UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE INGENIERÍA INDUSTRIAL

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

TESIS

"DESARROLLO DE EASYMAPS, UNA LIBRERÍA EN JAVASCRIPT PARA APLICACIONES WEB CON MAPAS QUE USEN OPENLAYERS Y GEOSERVER Y QUE UTILICEN LOS ESTÁNDARES WMS, WFS Y WPS"

PRESENTADA POR:

Bach. Henry Vanner Aquino Vegas

ASESORADA POR:

Msc. Carmen Zulema Quito Rodríguez

Piura, Perú

2019

UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE INGENIERÍA INDUSTRIAL

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

TESIS

"DESARROLLO DE EASYMAPS, UNA LIBRERÍA EN JAVASCRIPT PARA APLICACIONES WEB CON MAPAS QUE USEN OPENLAYERS Y GEOSERVER Y QUE UTILICEN LOS ESTÁNDARES WMS, WFS Y WPS"

Firmas de ejecutores:

Responsable:

Bach. HENRY VANNER AQUINO VEGAS

Asesor:

MSc. CARMEN ZULEMA QUITO RODRIGUEZ

Piura, Perú

2019

DECLARACIÓN JURADA

DECLARACIÓN JURADA DE ORIGINALIDAD DE LA TESIS

Yo: **AQUINO VEGAS HENRY VANNER**, identificado con **DNI Nº 74243367**, egresado de Facultad De Ingeniería Industrial, Escuela Profesional De Ingeniería Informática, domiciliado en Jr. Ramón Castilla 210, del Distrito de Catacaos, Provincia de Piura, Departamento de Piura.

Celular: 945481154.

Email: **henryvanner@gmail.com**

"DESARROLLO DE EASYMAPS, UNA LIBRERÍA EN JAVASCRIPT PARA APLICACIONES WEB CON MAPAS QUE USEN OPENLAYERS Y GEOSERVER Y QUE UTILIcEN LOS ESTÁNDARES WMS, WFS Y WPS"

DECLARO BAJO JURAMENTO: que la tesis que presento es original e inédita, no siendo copia parcial ni total de una tesis desarrollada, y/o realizada en el Perú o en el Extranjero, en caso contrario de resultar falsa la información que proporciono, me sujeto a los alcances de lo establecido en el Art. Nº 411, del código Penal concordante con el Art. 32º de la Ley Nº 27444, y Ley del Procedimiento Administrativo General y las Normas Legales de Protección a los Derechos de Autor. En fe de lo cual firmo la presente.

Piura, 23 de octubre del 2019

.....

DNI Nº 74243367

Artículo 411.- El que, en un procedimiento administrativo, hace una falsa declaración en relación con hechos o circunstancias que le corresponde probar, violando la presunción de veracidad establecida por ley, será reprimido con pena privativa de libertad no menor de uno ni mayor de cuatro años.

Art. 4. Inciso 4.12 del Reglamento del Registro Nacional de Trabajos de Investigación para optar grados académicos y títulos profesionales –RENATI Resolución de Consejo Directivo Nº 033-2016-SUNEDU/CD

UNIVERSIDAD NACIONAL DE PIURA
FACULTAD DE INGENIERÍA INDUSTRIAL
DECANATO

ACTA DE EVALUACIÓN Y SUSTENTACIÓN DE TESIS

Expediente N° 1542 / 2017

Los miembros del Jurado Calificador Ad-Hoc de la Sustentación de Tesis nombrado con Resolución N° **023-CF-FII-UNP-18** de fecha 11/01/2018 que suscriben, se reunieron en acto público en la sala de exposiciones de la Facultad de Ingeniería Industrial de la Universidad Nacional de Piura, el día **18 de Octubre del 2019** a las **11:00 am**, para evaluar la defensa de la Tesis titulada **"DESARROLLO DE EASYMAPS, UNA LIBRERÍA EN JAVASCRIPT PARA APLICACIONES WEB CON MAPAS QUE USEN OPENLAYERS Y GEOSERVER Y QUE UTILICEN LOS ESTÁNDARES WMS, WFS, WPS"**, presentada por el Bachiller **HENRY VANNER AQUINO VEGAS** y asesorado por la **MSc. CARMEN ZULEMA QUITO RODRÍGUEZ** .

Después de haber calificado el Informe Final de la Tesis, escuchada la sustentación y las respuestas a las preguntas formuladas por el Jurado, se le declara APROBADA para optar el Título de **INGENIERO INFORMÁTICO** con el puntaje de 81.9 que corresponde al calificativo de SOBRESALIENTE

Jurado / Calificación	Presidente	Secretario	Vocal	Puntaje Promedio
Documento (Max 60 puntos)	55	39	60	51.3
Sustentación (Max 40 puntos)	35	24	33	30.6
PUNTAJE TOTAL				81.9

En consecuencia, el sustentante queda en condición de recibir el Título Profesional que se indica, conferido por el Consejo Universitario de la Universidad Nacional de Piura de conformidad con las Normas Estatutarias y la Ley Universitaria en vigencia.

Ciudad Universitaria, 18 de Octubre del 2019

MSc. HÉCTOR WILMER FIESTAS BANCAYÁN	MBA. PERSI WILLIANSH CABRERA ANTON	MSc. ESTHER YOLANDA LIZANA PUELLES
PRESIDENTE	**SECRETARIO**	**VOCAL**

UNIVERSIDAD NACIONAL DE PIURA

FACULTAD DE INGENIERÍA INDUSTRIAL

ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA

TESIS

"DESARROLLO DE EASYMAPS, UNA LIBRERÍA EN JAVASCRIPT PARA APLICACIONES WEB CON MAPAS QUE USEN OPENLAYERS Y GEOSERVER Y QUE UTILICEN LOS ESTÁNDARES WMS, WFS Y WPS"

Aprobador por:

MSc. HÉCTOR WILMER FIESTAS BANCAYÁN
PRESIDENTE

MSc. ESTHER YOLANDA LIZANA PUELLES
VOCAL

MBA. PERSI WILLIANSH CABRERA ANTÓN
SECRETARIO

DEDICATORIA

Dedico este trabajo, como todos mis logros, a mi familia, que son las personas más importantes de mi vida. En especial a mis padres, con quienes estaré eternamente agradecido por el apoyo brindado y las lecciones impartidas, y por su compañía y amistad que me reconfortan.

AGRADECIMIENTOS

Agradezco en primer término, a la Ingeniera Carmen Zulema Quito Rodríguez, por su apoyo y guía constante en la elaboración del presente trabajo de investigación, y sus conocimientos y orientaciones que han sido fundamentales para guiarme en el camino de la investigación.

Agradezco también a los colegas que me apoyaron y a todas las personas que de una u otra forma han hecho posible la realización de este trabajo, puntualmente al Ing. Arturo Sandoval Rivera, por las facilidades prestadas y la oportunidad brindada que me llevó a conocer el fascinante mundo de los sistemas de información geográfica.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

El presente trabajo de investigación, tuvo como objetivo el desarrollo de una librería basada en Openlayers, para aplicaciones web de mapas, que implemente funcionalidades adicionales relacionadas a los servicios WMS, WFS y WPS de GeoServer.

Para el logro del objetivo se llevó a cabo una exploración de características de 53 visores web de mapas, de la cual se obtuvo una lista de funcionalidades comunes a ellos, que sirvió de base para la especificación de la lista de requerimientos que debía cumplir. Usando el lenguaje unificado de modelado (UML), se elaboró un modelo de la librería que cumple con los requerimientos especificados, y posteriormente, se ejecutó el desarrollo (codificación) de dicha librería según el modelo especificado, conjuntamente con la elaboración de la documentación web para su uso.

Una vez terminada la librería, esta se sometió a evaluación a través de un método basado en el estándar ISO/IEC 25000 de calidad del software, en la que se consideraron tres características a evaluar: funcionalidad (cumplimiento de los requerimientos), compatibilidad (con Openlayers) y usabilidad (simplificación del uso de Openlayers); esta última en relación a la creación de estilos para features vectoriales y el dibujo de features.

Los resultados obtenidos demostraron que la librería cumple con los requerimientos y es compatible con Openlayers, y que además, ofrece en mayor grado, una forma de uso simplificada para la creación de estilos para features vectoriales, y en menor grado, una forma de uso simplificada para el dibujo de features.

ABSTRACT

This work has been done, having had for main purpose, the development of a library based on Openlayers, for web mapping applications, which implements functionality related to GeoServer WMS, WFS and WPS services.

To achieve this goal, an exploration over 53 web mapping viewers was performed, obtaining as a result a list of common functionalities that were took into account to prepare the requirements specification the library had to fulfil. Using the Unified Model Language (UML), a model of the library that fulfil the requirements was made, and afterwards, the library was developed (coded) based on that model.

Once coding was finished and library was ready, this was evaluated using a method for the software quality evaluation based on ISO/IEC 25000, and three characteristics were considered: functionality (the library fulfills the requirements), compatibility (the library and Openlayers work well together) and usability (the library simplifies the use of Openlayers); the last one related to the creation of styles for vector features and drawing of features.

Results obtained showed that the library fully meets the requirements and it is compatible with Openlayers, and also, that it offers a significant simplification for the creation of styles for vector features, and a minor one, for drawing features.

Key Words: Librería, JavaScript, Openlayers, GeoServer, Aplicaciones web de mapas, estándares WMS, WFS y WPS.

INTRODUCCIÓN

En el desarrollo de aplicaciones web de mapas, los estándares Web Map Service (WMS), Web Feature Service (WFS) y Web Processing Service (WPS), cumplen el rol de un conjunto de especificaciones que norman la forma en que por medio de la web, los mapas deben ser creados, la información geográfica debe ser compartida y el procesamiento de la información debe ser ejecutado; que han sido adoptados por servidores de mapas como GeoServer, que implementa la mayoría de las operaciones que estos estándares definen; mientras que librerías como Openlayers, soportan la creación de mapas interactivos en base a mapas generados por servidores WMS como GeoServer.

Sin embargo, Openlayers cuenta con un número limitado de funciones relacionadas con operaciones de dichos estándares, y parte de dichas funciones están presentes en diferentes aplicaciones web de mapas, tal que, ante esta ausencia y la eventual y muy probable necesidad de contar con dichas funciones para el desarrollo de una nueva aplicación web, los programadores se ven en la necesidad de desarrollarlas por su cuenta, situación, que es incompatible con el paradigma de reutilización de software para agilizar el desarrollo.

Ante este panorama, se desarrolló este trabajo de investigación, con el objetivo de dar solución al problema referido, por medio de la creación de una librería que complemente a Openlayers, y que incluya funcionalidades relacionadas con los estándares WMS, WFS y WPS, para que opere con GeoServer. El trabajo contempla la evaluación de la calidad de la librería al final de su desarrollo (codificación) y la elaboración de su respectiva documentación.

CAPÍTULO I. ASPECTOS DE LA PROBLEMÁTICA

1.1.DESCRIPCIÓN DE LA REALIDAD PROBLEMÁTICA

Desde el lanzamiento de su versión 3.0 (en agosto del año 2014), Openlayers ha venido agregando más funcionalidades, cubriendo así, cada vez más las necesidades que se presentan en el desarrollo de aplicaciones web de mapas. No obstante, y a pesar de las significativas mejoras que se introdujeron en su última versión, esta librería aun cuenta con pocas funcionalidades relevantes relacionadas con los estándares WMS, WFS y WPS, desaprovechando así, sus capacidades. Sin embargo, cabe señalar que viene ampliando paulatinamente su cobertura sobre dichos, más dichas funcionalidades no están prestas para su uso, pues antes, es necesaria la creación y combinación de algunos objetos. (por ejemplo, la funcionalidad *GetFeatureInfo* de WMS).

Aunado a esto, Openlayers, ha aumentado la complejidad de algunas de sus funcionalidades en aras de adicionarles mayores cualidades y proveer una mejor normalización. Esto, no obstante, significó complicar su utilización incluso para fines sencillos. Un ejemplo puntual, es el de la creación de un estilo que define el relleno (color) y borde (ancho y color) para un polígono, donde pasó, de uno a tres, el número de instancias de clases necesarias para dicho fin; incrementando incluso a cuatro, si es que se desea añadir un texto al polígono.

Openlayers sigue siendo, y quizás ahora aún más, una librería con una API de grandes dimensiones (número de objetos, métodos y propiedades), lo que le otorga una gran flexibilidad y un mayor número de configuraciones. Sin embargo, como señala Fernandes (2012), esto puede tornarse en una barrera para la productividad de los programadores, viéndose afectada en varios niveles: la curva de aprendizaje de la API, la facilidad de memorización de las llamadas a las funcionalidades de la API (con posible consecuencia de errores o fallas de interpretación que los programadores hacen al utilizarlas) y en la forma como los programadores comprenden la API.

Se plantea entonces la pregunta:

¿Cómo desarrollar una librería en JavaScript, basada en Openlayers, para aplicaciones web con mapas, que incluya, funcionalidades relacionadas a los estándares WMS, WFS y WPS, y que trabaje con GeoServer?

1.2. JUSTIFICACIÓN E IMPORTANCIA DE LA INVESTIGACIÓN

1.2.1. Justificación

Ante esta situación de ausencia de funcionalidades relacionadas a los estándares WMS, WFS y WPS, o de funcionalidades que no están prestas para su uso, los programadores, se ven en la necesidad de invertir tiempo y esfuerzo en su desarrollo o configuración para su uso. Dichas funcionalidades, además, son básicas y fundamentales tal que se puede advertir que la mayoría de aplicaciones web de mapas las utilizan, lo que implica, que su desarrollo o configuración, se convierte en una tarea innecesariamente insoslayable, lo cual no va acorde con el paradigma de reutilización de software (creación de software a partir de software existente). Por consiguiente, el presente trabajó de investigación, ayudará a suplir la ausencia y satisfacer la necesidad manifiesta de una herramienta que incluya funcionalidades relacionadas con los estándares WMS, WFS y WPS, con el consecuente impacto positivo de agilizar el desarrollo de aplicaciones web de mapas, desarrolladas con Openlayers, y que trabajen con GeoServer.

Por otra parte, se pretende que la librería, también facilite el uso de algunas funcionalidades de Openlayers cuando se les requiere para un fin sencillo (como el ejemplo de la creación de un estilo mencionado anteriormente), tal como lo fueron en su versión anterior, con una API simplificada. Lo que, sumado a lo anterior, y como lo confirma Fernandes (2012), contribuirá positivamente en la productividad de los programadores.

1.2.2. Importancia

El presente trabajo de investigación, constituye posiblemente de los primeros (sino el primero), que aborda la temática de inclusión de funcionalidades relacionadas a los estándares WMS, WFS y WPS en una librería para aplicaciones de mapas, puntualmente, Openlayers. Asimismo, ofrece una solución a la mencionada carencia, con la creación de una nueva librería que incorpora funcionalidades que aprovechan las capacidades de dichos estándares, específicamente de las implementaciones del servidor de mapas GeoServer, y que se integra perfectamente con Openlayers.

Por otro lado, con el presente trabajo de investigación, se pretende promover el aprendizaje e investigación que concierne a la teoría y las tecnologías involucradas en el desarrollo de aplicaciones web de mapas, fomentar su uso para la creación de aplicaciones que saquen provecho de la información espacial, así como la creación de nuevas herramientas que susciten el desarrollo de dichas aplicaciones y la evolución de las tecnologías existentes.

1.3.OBJETIVOS

1.3.1. Objetivo general

Desarrollar una librería en JavaScript basada en Openlayers, para aplicaciones web de mapas, que implemente funcionalidades relacionadas con los servicios WMS, WFS y WPS de GeoServer.

1.3.2. Objetivo específicos

- Definir las funcionalidades (requerimientos del software) que serán incluidas en la librería.
- Elaborar un modelo de la librería que cumpla con los requerimientos.
- Desarrollar la librería según en el modelo creado.
- Elaborar la documentación para el uso de la librería.
- Someter la librería a evaluación.

1.4.DELIMITACIÓN DE LA INVESTIGACIÓN

1.4.1. Delimitación de la aplicación

La librería se desarrolló en base a versiones específicas de Openlayers y GeoServer, de modo que, es originalmente compatible con dichas versiones. Sin embargo, esto no significa que la librería no sea compatible con versiones más actuales. Significa, que la librería se desarrolló en base a las funcionalidades, APIs, especificaciones, y otras características de dichas versiones, por lo que es inherentemente compatible con ellas.

La compatibilidad de la librería con versiones más actuales de Openlayers y GeoServer, es posible en cuanto no haya cambios en sus funcionalidades, APIs, especificaciones, y otras características, que provoquen el mal funcionamiento de la librería. Cabe mencionar, que Openlayers y GeoServer, son proyectos considerados en una etapa adulta de su desarrollo, donde los cambios drásticos son poco probables.

Ítem	**Versión**
Openlayers	5.3.2
GeoServer	2.10.0

Tabla 1. Versiones de GeoServer y Openlayers en que está basada la librería.

Fuente: Elaboración propia

1.4.2. Delimitaciones de la evaluación

Para la evaluación de la calidad de la librería se utilizó el método propuesto por Baldeón Villanes (2015) basado en el estándar ISO/IEC 25000, y se consideraron 3 sub-características de la calidad según el estándar: completitud funcional (de la característica ‘Funcionalidad’) y coexistencia e interoperabilidad (de la característica ‘Compatibilidad’). Esto fue así debido a la escasa y limitada información sobre las métricas que se puede encontrar en el documento que describe el método, y al elevado costo de adquisición del documento del estándar y sus divisiones, de un aproximado de 158 francos suizos por documento.

Las evaluaciones se realizaron con la participación de un profesional con experiencia en el desarrollo de aplicaciones web de mapas y dos profesionales con experiencia en desarrollo web que fueron capacitados para impartirles el conocimiento pertinente y enseñarles las herramientas necesarias a fin de que estuvieran aptos para evaluar la librería. Esto, dada la poca disponibilidad de profesionales especialistas o con experiencia el desarrollo de aplicaciones web de mapas.

CAPÍTULO II. MARCO TEÓRICO

2.1.ANTECEDENTES DE LA INVESTIGACIÓN

- Silvestre Fernandes (2012), Lisboa – Portugal; en su Tesis presentada para obtener el grado de Master en Ingeniería Informática, realizó un estudio comparativo entre tres interfaces de programación de aplicaciones (APIs) para mapas: GoogleMaps, ArcGIS y Openlayers. El estudio se llevó a cabo con el fin de permitir al desarrollador tomar una decisión más informada de la API que se ajuste a sus necesidades. Se pusieron a prueba las APIs en la implementación de diferentes prototipos que contemplan funcionalidades esenciales de aplicaciones web con mapas. Las medidas que se realizaron consideraron aspectos relacionados a la efectividad y usabilidad de las APIs. Los resultados muestran que la API de Openlayers es la que cuenta con un mayor número de propiedades y métodos, lo cual resulta en mayor flexibilidad, pero también mayor complejidad. Aunque la diferencia no fue mucha, Openlayers mostro un mayor número promedio de argumentos por función, estando relacionado este valor con mayor información que el desarrollador debe aprender. Este antecedente, además de mostrar que Openlayers es de las librerías más usadas para el desarrollo de aplicaciones web de mapas, son útiles para promover, en base a resultados cuantitativos, los esfuerzos dirigidos a simplificar el uso de Openlayers.
- Maynas Ferrer (2015), Cataluña –España; en su Tesis presentada para obtener el grado de Bachiller en Ingeniería Geomática y Topografía, realizó un estudio comparativo entre los diferentes servidores de mapas de código abierto: GeoServer, MapServer y MapGuide con el objetivo de evaluar el rendimiento de las implementaciones de los servicios WMS, WFS y Web Coverage Service (WCS) que estos servidores incluyen. Las pruebas que se realizaron consistieron en el envío de peticiones con extensión espacial aleatoria de los diferentes servicios implementados por cada servidor y se midió el tiempo de latencia, es decir el tiempo desde el momento en que se envía la solicitud hasta que se recibe la respuesta, pero además se hizo una comparación entre los servicios que ofrecen estos servidores. Los resultados de las pruebas llevaron a las siguientes conclusiones: (1) GeoServer y MapServer implementan más servicios, por ende, cubriendo mayores necesidades, quedándose MapGuide rezagado en este aspecto, el cual al tiempo de las pruebas no implementaba el servicio WFS. (2) GeoServer obtuvo los mejores resultados en rendimiento de las pruebas realizadas para el servicio WMS, y junto con MapServer obtuvieron las imágenes con mejor calidad; y (3) MapServer obtuvo el mejor rendimiento para las pruebas realizadas sobre el servicio WFS, sin embargo este resultado podría deberse a las diferencias en los contenidos de las respuestas de ambos servidores, siendo que GeoServer devuelve coordenadas XYZ de los elementos consultados del mapa, mientras que

MapServer solo mostraba las coordenadas XY y además que GeoServer ofrece una mejor organización de sus resultados. Este trabajo deja demostrado que tanto GeoServer como MapServer ofrecen un buen rendimiento al momento de ejecución de operaciones relacionadas a los estándares WMS y WFS, por tanto, son ideales para ser consideradas en el desarrollo de la librería referida en el presente trabajo. Sin embargo, por su mayor cobertura de operaciones de los estándares WMS y WFS, y porque soporta el estándar WPS gracias a una extensión, fue que se optó por GeoServer.

- Olaya Ordinola (2017), Piura – Perú; en su Tesis presentada para obtener el título de Ingeniero Informático, implementó un sistema web de información geográfica para la empresa prestadora de servicios de saneamiento Grau (EPS GRAU), para cuyo desarrollo se utilizó software libre, entre los cuales destacan Openlayers y GeoServer. El sistema fue desarrollado con el fin de integrar y gestionar la información técnico – comercial (tanto alfanumérica como geográfica) de la empresa; lo cual significó la implementación de funcionalidades para acciones tales como: registrar una nueva manzana, registrar un nuevo tramo de red, dividir un predio, entre otros. Funciones, dónde operaciones como el dibujo de figuras geométricas básicas (puntos, líneas y polígonos), entre otras; son esenciales y el sistema tenía que ser capaz de llevarlas a cabo. Por ejemplo, para el registro de una nueva manzana (la función), que implica el dibujo de un polígono. Dichas operaciones fueron implementadas con la ayuda de Openlayers, que también se usó para la visualización en el sistema, de mapas de redes, predios, usuarios, etc., que son generados por GeoServer gracias a su implementación del estándar WMS. El sistema también incluye funcionalidades básicas, pero que sin embargo resultan muy útiles, tales como el conteo de elementos (ya sean redes, predios, usuarios, etc.) dentro de un área específica, para cuya implementación fue necesario el desarrollo de operaciones con las que Openlayers no contaba, relacionadas con los estándares WMS, WFS y WPS. La implementación del sistema permitió mejorar la gestión de la información geográfica del catastro técnico-comercial de la empresa y mostró además una nueva forma de analizar dicha información basándose en relaciones espaciales entres los elementos. El desarrollo de dicho sistema fue base para la idea de crear una librería basada en Openlayers que cuente con funcionalidades relacionadas a los servicios WMS, WFS y WPS de GeoServer, tal que sirva para el desarrollo de sistemas con funcionalidades que se apoyan en dichos servicios.

2.2.BASES TEÓRICAS

2.2.1. Librería

Aunque quizás no exista una definición oficial u global de que es o comprende una librería (o biblioteca), se puede concebir o subsumir una idea del contenido de diferentes textos dónde este término es usado.

Por ejemplo, Microsoft (2019), señala sobre sus archivos DLL (Dynamic Link Library o Biblioteca de vínculos dinámicos), que son bibliotecas que contienen código y datos que pueden ser utilizados por varios programas al mismo tiempo. Por ejemplo, el archivo *Comdlg32.dll* realiza funciones comunes relacionadas con el cuadro de diálogo. Así, cada programa que use dicho archivo, puede invocar la funcionalidad contenida en él para implementar un cuadro de diálogo. Esto ayuda a promover la reutilización de código y suscita la división del programa en componentes separados, lo que afecta positivamente en el rendimiento y mantenimiento del programa.

Por otro lado, W. Kernighan & M. Ritchie (1988), señalan sobre la librería estándar que se especifica en el lenguaje C, que contiene un extenso conjunto de funciones para la entrada y salida de datos, administración de memoria, manipulación de *strings*, y tareas similares. La librería extiende el lenguaje con funcionalidades que el lenguaje en sí no contempla, por ejemplo, para la entrada y salida de datos, con ayuda de la función *printf*.

Brevemente Eckel (2000), declara que, la manera más fácil de crear un programa es usar código que ya está escrito: una librería.

Por su parte, Khan Academy (2019), en su artículo *'What's a JS library'*, indica que una librería es un archivo JavaScript que contiene un grupo de funciones que llevan a cabo algunas tareas útiles para páginas web.

Conforme a todo lo citado anteriormente, se puede decir, que una librería, es un archivo escrito en algún lenguaje, que provee un conjunto de funciones para la consecución de tareas específicas y que pueden ser reutilizadas por otros programas, de modo que facilitan su desarrollo. Cumplen con un rol complementario, pues proveen de capacidades que se necesitan, pero con las que en un principio no se cuentan. Y aunque no se menciona de manera explícita, una librería por su naturaleza, no cuenta con una interfaz gráfica de usuario (GUI), sino, con una interface de programación de aplicaciones (API), con la que el programador puede invocar las funcionalidades con las que la librería cuenta.

2.2.2. Digital Mapping, Web Mapping y Aplicaciones Web de Mapas

De manera concreta, *digital mapping* es el proceso de creación de mapas digitales, que surgió con la aparición de las computadoras e internet y la consecuente transformación digital del mundo.

En contraste, con los métodos tradicionales, el poder de digital mapping ha demostrado ser superior. El proceso convencional de creación de mapas incluye dibujos hechos a mano sobre papel de elementos del mundo real. Si algún elemento cambia, se mueve, o es dibujado incorrectamente, un nuevo mapa debe ser creado para reflejar tal cambio, tarea que resulta lenta y laboriosa.

Este y otros problemas se reducen con digital mapping, ya que los elementos son almacenados en archivos y a partir de ellos se crea un mapa. De modo que, si un elemento es modificado, el mapa instantáneamente refleja ese cambio la siguiente vez que el elemento es visualizado. A esto, se suman las virtudes de los mapas interactivos, que permiten al usuario ver el área específica en la cual están interesados (en lugar de estar limitados a las dimensiones de una página impresa) y elegir ver solo ciertas partes del contenido.

El creador de mapas digitales no se enfoca tanto en los detalles de un área particular del mundo, sino, en como presentar la información geográfica de la mejor manera para una audiencia específica. En el mundo digital, la atención se centra en ayudar a otros a encontrar la información que necesitan, en lugar de presentar la información de manera estática en una página impresa. Por eso, el creador de mapas de hoy, es a menudo un desarrollador de sitios web, un programador, o algún tipo de analista de información geográfica.

Una forma efectiva de poner información geográfica a disposición de un grupo de usuarios no técnicos es mostrándola en un sitio web. Sitios web de este tipo son denominados *web mapping sites*, de los que existen (en un sentido general) dos tipos: estáticos e interactivos.

Los mapas estáticos que se muestran como imágenes en sitios web son los más comunes. Con una imagen digital de un mapa (por ejemplo, la obtenida del escaneo de un documento), se puede rápidamente contar con un mapa estático en un sitio web. Solo se necesitan conocimientos básicos de diseño web.

Por otra parte, los mapas interactivos no son tan comunes, pues se requiere de habilidades especializadas para el mantenimiento de dichos sitios. El término interactivo implica que el usuario puede interactuar con el mapa del algún modo. Esto puede ser, seleccionando diferentes capas de datos para ver o hacer zoom en una parte específica del mapa que sea de interés. Todo esto mientras se interactúa con la página web y una imagen de un mapa que es constantemente actualizada para mostrar el contenido que el usuario requiere.

Los mapas interactivos son accedidos a través de páginas web conocidas como *mapas basados en la web* (web-based maps) o simplemente *mapas web*. Estos mapas, pueden ser muy potentes, pero como se mencionó anteriormente, pueden ser difíciles de configurar debido a las habilidades técnicas requeridas para mantener un servidor web, un servidor de mapas y para administrar los datos geográficos subyacentes. Estos tipos de mapas son fundamentalmente diferentes de los mapas estáticos porque son un tipo de aplicación web, son *aplicaciones web de mapas*. Para ver uno de estos mapas, el usuario hace una petición del mapa al servidor web, y el servidor pasa la petición al servidor de mapas, quien genera el mapa a partir de los datos geográficos. El mapa, es entonces pasado de vuelta al usuario y se muestra en el navegador web. (Mitchell, 2005)

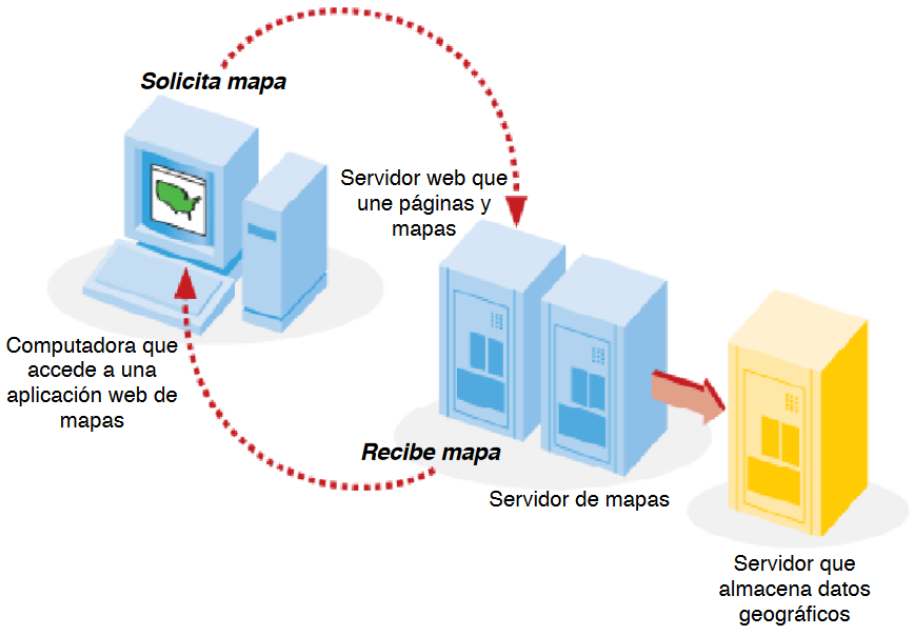


Figura 1. Diagrama de como interactúa un sitio web de mapas con el usuario y los servidores que están detrás de su funcionamiento.

Fuente: (Mitchell, 2005)

Elaboración propia

2.2.3. Información geográfica

Es aquella en cuyo significado incorpora una referencia a una posición en la superficie de la Tierra. En líneas generales, esta se puede dividir en dos componentes principales:

- Componente espacial
- Componente temática

La componente espacial hace referencia a la posición dentro de un sistema de coordenadas de referencia (SRS o CRS) establecido. Esta componente es la que hace que la información pueda calificarse como geográfica.

Por su parte, la componente temática es la que describe la naturaleza y características del fenómeno que tiene lugar en la ubicación que queda establecida por la componente espacial. (Olaya, 2014)

2.2.4. Mapa

Un mapa es el medio de comunicación por excelencia para transmitir la información geográfica de modo visual. De la misma forma que al hablar pretendemos transmitir algo y para ello usamos el lenguaje como herramienta, en el caso de crear un mapa empleamos el lenguaje gráfico para transmitir una determinada información geográfica. (Olaya, 2014)

Los mapas pueden ser las típicas representaciones gráfica en una hoja – como los mapas tradicionales –, pero también los son, los mapas digitales y sistemas de visualización terrestre. Después de todo, un mapa es simplemente una metáfora de la tierra. En un sentido más amplio, el término mapa incluye todas las metáforas de la tierra, desde tradicionales mapas cartográficos hasta sistemas de visualización 3D. Existen incluso mapas especiales de los que podemos aprender acerca de fenómenos que varían con el tiempo. (Open Geospatial Consortium [OGC], 2009)

2.2.5. Capas

Las capas son el resultado de la división de la información espacial referida a una zona de estudio en varios niveles en función de la componente temática.

Para comprender mejor el concepto de capa, es útil pensar en un mapa cartográfica clásico con elementos como curvas de nivel, carreteras, núcleos urbanos y sitios de interés arquitectónico (iglesias, monumentos, etc.) Todos estos elementos en su conjunto componen el mapa, y aparecen en una misma hoja como una unidad coherente de información geográfica. No obstante, cada uno de estos grupos de información – elevaciones, red vial, núcleos urbanos y sitios de interés arquitectónicos – pueden

recogerse de forma independiente, y combinarse para componer el mapa o bien emplearse individualmente. (Olaya, 2014)

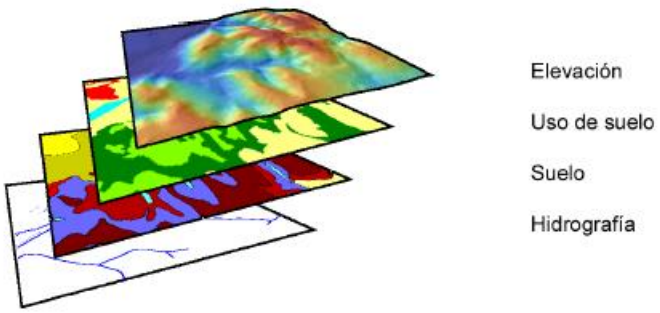


Figura 2. Concepto de capa de información geográfica.

Fuente: (Olaya, 2014)

2.2.5.1. Feature

Según el OGC (2009), Un Feature es la unidad fundamental de la información espacial. Es una abstracción de un fenómeno del mundo real.

Dependiendo de la aplicación o intereses de quien recolecta la información, un feature podría ser cualquier de los ítems que se muestran en la siguiente lista:

- Un segmento de un camino entre intersecciones consecutivas.
- Una carretera enumerada que consiste en muchos segmentos de caminos.
- Un camino segmentado dinámicamente.
- Una imagen satelital georreferenciada.
- Un pixel de la imagen mencionada anteriormente.
- Una red de drenaje.
- La sobre-cobertura de temperatura en un mapa climático.
- Un conjunto de contornos de magnitud de eventos sísmicos.

2.2.5.2. Modelos geográficos

Según Olaya (2014), el modelo geográfico es un modelo conceptual de la realidad geográfica y su comportamiento. Es un esquema mental que constituye una forma particular de entender el hecho geográfico.

Existen muchos modelos geográficos distintos, entre los cuales cabe destacar dos de ellos:

- Campos
- Entidades discretas

2.2.5.2.1. Campos

Un campo es un modelo de variación dentro de un marco n-dimensional, en el cual en cada punto dentro de dicho marco se tiene un valor de la variable estudiada.

Es un concepto matemático, un campo es una función de la forma f: $R^n \rightarrow R^m$, esto es, una función que asocia cada punto de un espacio vectorial con otro en un espacio vectorial distinto.

En el caso más habitual, cuando m = 1, es decir, que a cada punto del espacio vectorial origen se le asocia un único valor escalar, se obtiene lo que se denomina un campo escalar. La mayoría de las variables que se emplean en los sistemas informáticos necesitan un único valor para describirse (piénsese en variables como la elevación, la temperatura o la presión atmosférica, que solo requieren de un número para expresarse).

El espacio vectorial de origen puede ser bidimensional, es decir, una función de la forma $f(x, y)$, representando x e y las coordenadas geográficas. Este es el caso habitual en las capas, donde las variables que estudiamos adquieren uno u otro valor en función de su posición dentro de un sistema de coordenadas de referencia. (Olaya, 2014)

2.2.5.2.2. Entidades discretas

A diferencia de los campos, el modelo de entidades discretas no asocia a cada punto geográfico un valor, sino que concibe un entorno geográfico como un espacio vacío sobre el que se sitúan distintos elementos (entidades) que lo van rellenando.

Cada una de dichas entidades posee unas características propias, que son las que conferirán sus propiedades particulares a los puntos que si sitúen en su interior.

La presencia de vías de comunicación, por ejemplo, se puede asimilar perfectamente a este modelo. Se tiene un espacio vacío, en el cual se disponen las distintas vías en una serie de localizaciones concretas. (Olaya, 2014)

2.2.5.3. Modelos de representación

los modelos de representación, también llamados modelos de datos, son la forma de recoger la realidad que los modelos geográficos describen, reduciendo sus propiedades a un conjunto finito de

elementos. Estas formas de representación se pueden clasificar en dos grupos principales: modelo de representación ráster y modelo de representación vectorial.

Cabe señalar que, dado que los modelos ráster y vectorial son los principales empleados para la definición de capas, se utilizan las expresiones capa ráster y capa vectorial para señalar el tipo de modelo de representación usado. (Olaya, 2014)

2.2.5.3.1. Modelo ráster

El modelo ráster se basa en una división sistemática del espacio (a este concepto se le denomina teselación) en una serie de unidades mínimas (denominadas habitualmente celdas), y para cada una de estas se recoge la información pertinente que la describe.

El formato ráster es especialmente adecuado para el análisis de la información geográfica, en especial cuando esta es de tipo continuo (relieve, temperatura, etc.) Es más próxima al modelo geográfico de campos que al de entidades discretas.

No obstante, ello no restringe el alcance del formato. Variables que no resultan tan bueno concebir como campos, tales como una red vial, también pueden expresarse como una capa ráster. (Olaya, 2014)

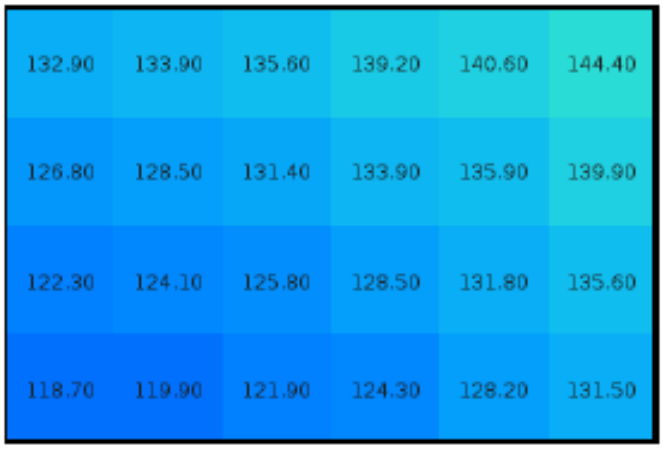


Figura 3. Celdas de una malla ráster con sus valores asociados.

Fuente: (Olaya, 2014)

2.2.5.3.2. Modelo vectorial

El modelo vectorial no divide el espacio, sino que lo define mediante una serie de entidades geométricas con valores asociados, siendo la disposición de estos no sistemática, sino guardando relación con los objetos geográficos presentes en la zona de estudio. La forma de estas entidades (su frontera), se

codifica de modo explícito, a diferencia del modelo ráster, donde viene implícita en la propia estructura de malla.

El modelo vectorial se asemeja al modelo geográfico de entidades discretas, pues modeliza el espacio geográfico mediante una serie de primitivas geométricas que contienen los elementos más destacados de dicho espacio. Estas primitivas son de tres tipos: puntos, líneas y polígonos. La elección de uno u otro tipo de primitiva para modelar una entidad y registrar una variable o conjunto de ellas ha de ser en función del tipo de fenómeno que se pretende modelizar o la precisión necesaria, entre otros factores.

Por ejemplo, una capa de puntos puede representar un conjunto de ciudades, cada una de ellas definida como un único punto. Sin embargo, puede emplearse una capa de polígonos y no recoger una única coordenada (correspondiente, por ejemplo, al centro de la ciudad), sino el contorno o los límites administrativos de esta. Dependiendo del caso, será más apropiado elegir una u otra alternativa. (Olaya, 2014)

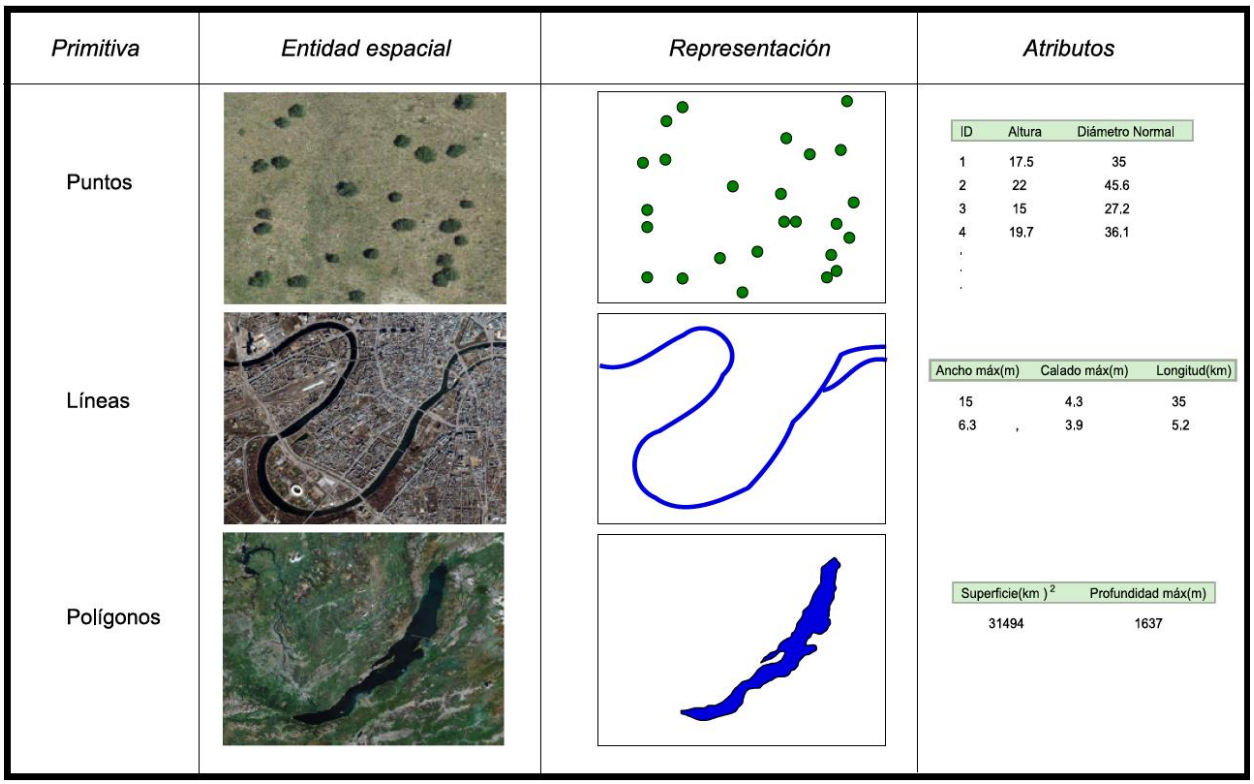


Figura 4. Primitivas geométricas en el modelo de representación vectorial y ejemplos particulares de cada una de ellas con atributos asociados.

Fuente: (Olaya, 2014)

2.2.5.4. Formatos para el almacenamiento de la información geográfica

Existen muchos formatos para la codificación de información geográfica, entre los cuales cabe destacar (por sus características que lo hacen idóneos para manejar información geográfica en JavaScript) GeoJSON y WKT.

2.2.5.4.1. GeoJSON

Es un formato basado en JSON (JavaScript Object Notation). Define varios tipos de objetos JSON y la forma en que se combinan para representar datos de features, sus propiedades (componente temática), y su extensión espacial (componente geográfica).

Con los objetos GeoJSON se pueden representar, una región del espacio (una geometría), una entidad delimitada espacialmente (un feature), o una lista de features (Colección de features). Soporta además los siguientes tipos de geometría: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. (Internet Engineering Task Force, 2019)

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-80.626675,-5.196772]
  },
  "properties": {
    "nombre": "Plaza de armas de Piura"
  }
}
```

Figura 5. Ejemplo de codificación de feature usando el formato GeoJSON.

Fuente: Elaboración propia

2.2.5.4.2. Well Known Text (WKT)

WKT es un estándar del OGC usado para la representación de datos espaciales en un formato textual. Una expresión WKT solo puede almacenar la información de un único objeto espacial (componente geográfica). Sin embargo, dado que es nada más que una cadena de caracteres con un formato definido, puede ser fácilmente creado y decodificado. La gran mayoría de los sistemas que usan los estándares del OGC soportan este estándar. WKT soporta los siguientes tipos de geometría: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. (Microsoft, 2019)

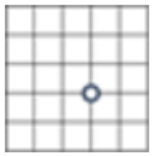
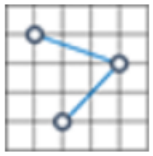
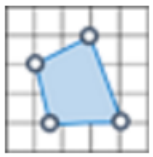
Tipo de geometría		Ejemplo
Point		POINT(-122.349 47.651)
LineString		LINESTRING(-122.360 47.656, -122.343 47.656)
Polygon		POLYGON((-122.358 47.653, -122.348 47.649, -122.348 47.658, -122.358 47.658, -122.358 47.653))

Figura 6. Objetos espaciales codificados usando el formato WKT.

Fuente: (Microsoft, 2019)

Elaboración propia

2.2.6. Web Map Service (WMS)

El servicio WMS, es un estándar del OGC que define una interface para producir mapas usando el protocolo HTTP. Aquí, un mapa se define como una representación visual de datos geográficos, mas no como los datos en sí.

El estándar, define algunas operaciones a ser implementadas por los fabricantes que deseen cumplirlo; algunas son obligatorias, mientras que otras, opcionales. La invocación de dichas operaciones se realiza con el uso de URLs; estas (las operaciones), aceptan parámetros cuyos valores determinan la forma del resultado de su ejecución. (OGC, Web Map Service Implementation Specification, 2002)

De las operaciones definidas por este estándar, cabe destacar tres de ellas:

- GetMap
- GetFeatureInfo
- GetLengendGraphic

2.2.6.1. GetMap

Esta operación está diseñada para producir un mapa, el cual, puede ser una imagen o un conjunto de elementos gráficos; su implementación es obligatoria. (OGC, Web Map Service Implementation Specification, 2002)

A continuación, se describen los parámetros más relevantes de esta operación.

Parámetro	Descripción
SERVICE	Nombre del servicio. ('WMS' en este caso)
VERSION	Versión del estándar.
REQUEST	Nombre de la operación. ('GetMap' en este caso)
LAYERS	Lista de capas que serán incluidas en el mapa. (separadas por comas)
STYLES	Lista de estilos que serán usados para las capas, uno por cada capa. (separados por comas)
SRS	Sistema de Coordenadas de Referencia.
BBOX	Porción de la tierra que abarcará el mapa.
WIDTH	Ancho en píxeles de la imagen.
HEIGHT	Altura en píxeles de la imagen.
FORMAT	Formato de salida del mapa. Por ejemplo, GIF, PNG, JPEG, etc.
EXCEPTIONS	Formato en el cual las excepciones deberán ser reportadas.
Parámetros específicos del fabricante.	Parámetros experimentales opcionales que pueden ser agregados por el fabricante.

Tabla 2. Parámetros de una petición GetMap.

Fuente: (OGC, Web Map Service Implementation Specification, 2002).

Elaboración propia

2.2.6.2. GetFeatureInfo

Esta operación está diseñada para proveer mayor información acerca de los features de las imágenes de mapas que son retornadas por peticiones GetMap. El uso canónico de esta operación, es que un usuario primero ve la respuesta de una petición GetMap y luego elige un punto en el mapa del cuál obtener mayor información. A través de sus parámetros, esta operación provee al cliente, la habilidad de

especificar el pixel que se va a consultar, las capas que deberían ser investigadas, y el formato en el que la información debería ser retornada. Su implementación es opcional. (OGC, Web Map Service Implementation Specification, 2002)

A continuación, se describen los parámetros más relevantes de esta operación:

Parámetro	Descripción
SERVICE	Nombre del servicio. ('WMS' en este caso)
VERSION	Versión del estándar.
REQUEST	Nombre de la operación. ('GetFeatureInfo' en este caso)
<parámetros_petición_getMap>	Copia parcial de los parámetros de la petición GetMap que generó el mapa del cual se desea información.
INFO_FORMAT	Formato en que se debe retornar la información.
FEATURE_COUNT	Número de features de los cuales retornar información.
X	Coordenada X del feature, en píxeles. (medido desde la esquina superior izquierda)
Y	Coordenada Y del feature, en píxeles. (medido desde la esquina superior izquierda)
EXCEPTIONS	Formato en el cual las excepciones deberán ser reportadas.
Parámetros específicos del fabricante.	Parámetros experimentales opcionales que pueden ser agregados por el fabricante.

Tabla 3. Parámetros de una petición GetFeatureInfo.

Fuente: (OGC, Web Map Service Implementation Specification, 2002)

Elaboración propia

2.2.6.3. GetLegendGraphic

Esta operación provee un mecanismo para la generación de leyendas en forma de imágenes. Su implementación es opcional. (OGC, Styled Layer Descriptor profile of the Web Map Service Implementation Specification, 2007)

A continuación, se describen los parámetros más relevantes de esta operación:

Parámetro	Descripción
SERVICE	Nombre del servicio. (‘WMS’ en este caso)
VERSION	Versión del estándar.
REQUEST	Nombre de la operación. (‘GetLegendGraphic’ en este caso)
LAYER	Capa para la cual producir la leyenda.
STYLE	Estilo de la capa que se deberá usar para producir la leyenda.
FORMAT	Formato en el cual se debe retornar la leyenda.
WIDTH	Ancho de los íconos presentes en la leyenda, en píxeles.
HEIGHT	Altura de los íconos presentes en la leyenda, en píxeles.
EXCEPTIONS	Formato en el cual las excepciones deberán ser reportadas.

Tabla 4. Parámetros de una petición GetLegendGraphic.

Fuente: (OGC, Styled Layer Descriptor profile of the Web Map Service Implementation Specification, 2007)

Elaboración propia

2.2.7. Web Feature Service (WFS)

El servicio WFS, es un estándar que define interfaces para la ejecución de operaciones de acceso y manipulación de features usando el protocolo HTTP. A través de estas interfaces, un cliente web puede consultar, usar y administrar la información espacial a partir de la cual se producen los mapas. (OGC, Web Feature Service Implementation Specification, 2005)

De las operaciones definidas por este estándar, cabe destacar dos de ellas:

- GetFeature
- DescribeFeatureType

2.2.7.1. GetFeature

Esta operación, proporciona al cliente la habilidad de recuperar features, con opción de especificar que propiedades capturar y de restringir la consulta espacialmente o no-espacialmente. Es

decir, que un cliente web puede recuperar features según criterios basados en su ubicación o los valores de sus propiedades.

Para este estándar, un feature, es una instancia de un tipo de feature, que es la abstracción de un fenómeno del mundo real que tiene lugar en un espacio en la tierra. Para entenderlo mejor, piense en un tipo de feature como en una clase y un feature como un objeto (instancia de una clase).

Un servidor web de mapas puede responder a una petición GetFeature de una de dos formas; puede generar una respuesta completa que contiene todos los features que satisfacen la petición, o simplemente devolver el número de features que la petición retornaría. Su implementación es obligatoria. (OGC, Web Feature Service Implementation Specification, 2005)

A continuación, se describen los parámetros más relevantes de esta operación:

Parámetro	Descripción
SERVICE	Nombre del servicio. (‘WFS’ en este caso)
VERSION	Versión del estándar.
REQUEST	Nombre de la operación. (‘GetFeature’ en este caso)
outputFormat	Formato en el cual se debe retornar la información de los features.
resultType	Tipo de respuesta. ‘Results’ para retornar todos los features o ‘Hits’ para retornar solo el número de features.
maxFeatures	Máximo número de features que debe ser retornado.
typeName	Nombre de uno o más tipos de features que serán consultados.
srsName	Sistema de Coordenadas de Referencia a usar para la geometría de los features.
propertyName	Lista de propiedades de los features que deberán ser capturadas. (Separadas por comas)
filter	Restricción para los features que serán retornados.
sortBy	Orden en que debe ser retornada la información de los features, ascendente o descendente, según sus propiedades,

bbox	Extensión (Área rectangular) a la cual se debe limitar la consulta.

Tabla 5. Parámetros de una petición GetFeature.

Fuente: (OGC, Web Feature Service Implementation Specification, 2005)

2.2.7.2. DescribeFeatureType

La función de esta operación es proveer una descripción del esquema de los tipos de feature. El esquema, define la forma que tendrán las instancias de los tipos de feature (o solo features); por lo tanto, la forma que tendrán en la respuesta a una petición GetFeature. Su implementación es obligatoria. (OGC, Web Feature Service Implementation Specification, 2005)

A continuación, se describen los parámetros más relevantes de esta operación:

Parámetro	Descripción
SERVICE	Nombre del servicio. ('WFS' en este caso)
VERSION	Versión del estándar.
REQUEST	Nombre de la operación. ('DescribeFeatureType' en este caso)
outputFormat	Formato en el cual se debe retornar la descripción del esquema.
typeName	Nombre de uno o más tipos de features que serán descritos.

Tabla 6. Parámetros de una petición DescribeFeatureType.

Fuente: (OGC, Web Feature Service Implementation Specification, 2005)

2.2.8. Web Processing Service (WPS)

Es un estándar del OGC que provee al cliente acceso para la ejecución de cálculos pre-programados y/o modelos de cálculo aplicados a datos georreferenciados, usando el protocolo HTTP. Los cálculos pueden ser tan simples como la substracción de un conjunto de números georreferenciados de otro (por ejemplo, para determinar la diferencia en casos de influencia en dos temporadas diferentes), o tan complicados como un modelo del cambio climático global.

Ya que el procesamiento geoespacial en internet requiere el desarrollo de una amplia variedad de servicios web para soportar todas las operaciones geoespaciales posibles, WPS surge para estandarizar la forma en que estas operaciones son llamadas a fin de reducir la cantidad de codificación requerida, y para

facilitar la implementación y adopción de nuevos servicios. (OGC, OpenGIS Web Processing Service, 2007)

De las operaciones definidas por este estándar, cabe destacar la operación Execute.

2.2.8.1. Execute

Esta operación, permite al cliente ejecutar procesos específicos implementados por un servidor, usando los valores de parámetros proveídos y retornando los valores de salida producidos. Su implementación es obligatoria. (OGC, OpenGIS Web Processing Service, 2007)

Los parámetros de esta operación son:

Parámetro	Descripción
SERVICE	Nombre del servicio. ('WPS' en este caso)
VERSION	Versión del estándar.
REQUEST	Nombre de la operación. ('Execute' en este caso)
Identifier	Identificador del proceso.
DataInputs	Datos de entrada que se usarán en la ejecución del proceso.
ResponseForm	Tipo de respuesta,
language	Identificador de lenguaje. (RFC4646)

Tabla 7. Parámetros de una petición Execute.

Fuente: (OGC, OpenGIS Web Processing Service, 2007)

2.2.9. Geoserver

GeoServer es un servidor web de mapas basado en Java, que permite a los usuarios ver y editar datos geoespaciales. Usa estándares dados por la OGC para ofrecer gran flexibilidad en la creación de mapas y el intercambio de datos.

El servidor viene integrado con los estándares WMS y WFS. Una implementación del estándar WPS está disponible como una extensión. (OSGeo, 2019)

2.2.9.1. Web Map Service

GeoServer soporta las versiones 1.1.1 y 1.3.0 de este estándar. De las operaciones que implementa, cabe destacar las siguientes:

- GetMap

- GetFeatureInfo
- GetLegendGraphic

El servicio soporta varios formatos para la generación de mapas, de los cuáles cabe mencionar a PNG, JPEG y KML. En lo que concierne a la operación GetFeatureInfo, la información de los features se puede retornar en el formato GeoJSON.

Adicionalmente, se incluyen una variedad de parámetros específicos del fabricante que proveen capacidades mejoradas al servicio. (OSGeo, 2019)

De estos parámetros, cabe destacar los siguientes:

Parámetro	Descripción
cql_filter	Filtro expresado usando ECQL (Extended Common Query Language).
propertyName	Propiedades de los features que deben ser incluidas en la respuesta de una operación GetFeatureInfo.
tiled	Define en una operación GetMap, si el mapa se debe retornar como una única imagen o se debe dividir en teselas antes de ser enviadas al cliente.
LEGEND_OPTIONS	Permite un control más fino sobre la apariencia de la leyenda. Algunas de las propiedades que se pueden configurar son: • Tipo de fuente. • Tamaño, grosor y color de letra. • Color de fondo.

Tabla 8. Parámetros específicos del fabricante del servicio WMS de GeoServer.

Fuente: (OSGeo, 2019)

Elaboración propia

2.2.9.2. Web Feature Service

GeoServer soporta las versiones 2.0.0, 1.1.0 y 1.0.0 de este estándar.

El servicio soporta varios formatos de salida para la información, de los cuáles cabe mencionar a GML, Shapefile, GeoJSON y CSV.

Adicionalmente, se incluyen algunos ‘parámetros específicos del fabricante’ que proveen capacidades mejoradas al servicio. (OSGeo, 2019)

De estos parámetros, cabe destacar los siguientes:

Parámetro	Descripción
cql_filter	Filtro expresado usando ECQL (Extended Common Query Language).
format_options	Permite un control más fino sobre el formato de salida de la información.

Tabla 9. Parámetros específicos del fabricante del servicio WFS de GeoServer.

Fuente: (OSGeo, 2019). Elaboración propia

2.2.9.3. Web Processing Service

El servicio no viene integrado por defecto con GeoServer, sin embargo, está disponible como una extensión. El servicio WPS de GeoServer, provee una integración directa con otros servicios de GeoServer. Esto significa, que es posible ejecutar procesos sobre los datos que son servidos por GeoServer sin necesidad de enviar todos estos datos en la petición. Además, es posible guardar los resultados del procesamiento directamente en GeoServer.

Este servicio incluye algunos procesos creados específicamente por GeoServer, de los cuáles, cabe mencionar los procesos de **gs:Aggregate** y **gs:Bounds**. El primero, se usa para ejecutar funciones comunes de agregación (suma, promedio, máximo, etc.) sobre la información de los features. El segundo, para calcular el *bounding box* de un conjunto de features. El resultado de ambas operaciones puede ser retornado en formato XML o JSON (OSGeo, 2019)

A continuación, se describen los parámetros de ambas operaciones:

Parámetro	Descripción
features	Colección de features cuya información será procesada.
aggregationAttribute	Propiedad del feature sobre el cuál ejecutar la agregación.
function	Lista de funciones de agregación que se ejecutarán. Pudiendo incluir Count, Average, Max, Median, Min, StdDev, y Sum.
groupByAttributes	Propiedad a utilizar para la agrupación de los datos.

Tabla 10. Parámetros del proceso 'gs:Aggregate' del servicio WPS de GeoServer.

Fuente: (OSGeo, 2019)

Elaboración propia

Parámetro	Descripción
features	Colección de features.

Tabla 11. Parámetros del proceso 'gs:Bounds' del servicio WPS de GeoServer.

Fuente: (OSGeo, 2019)

Elaboración propia

2.2.10. Openlayers

Openlayers es una librería para mapas web hecha en JavaScript que ofrece componentes para trabajar con información geográfica en el navegador. Apareció en los mediados del 2006 como una alternativa de código abierto a Google Maps y otras APIs, pero empezó a ganar más atención en 2007, cuando el proyecto OpenStreetMap (entonces en desarrollo) lo adopta para su sitio web. Hoy en día, es un proyecto adulto y probablemente la más potente y completa librería para trabajar con información geográfica en el lado del navegador.

Openlayers está preparado para visualizar datos ráster y vectoriales en el navegador, y más que eso, soporta diferentes fuentes de datos, desde servidores de mapas que implementan estándares de la OGC hasta simples archivos planos en diferentes formatos: GML, GKML, GeoJSON, etc. Permite trabajar con datos en diferentes sistemas de coordenadas de referencia. Implementa el concepto de capa para colocar diferentes tipos de datos y mostrarlas u ocultarlas para ver solo aquellos datos que queremos. Permite dar estilo a los features dependiendo de sus atributos. (Santiago, 2015)

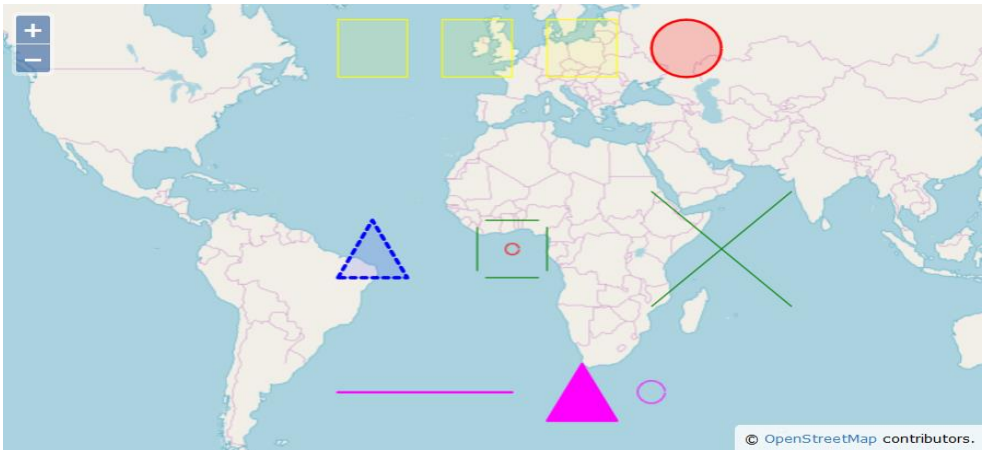


Figura 7. Ejemplo de mapa que muestra features con diferentes estilos.

Fuente: (Openlayers, 2019)

Elaboración propia

2.2.11. Javascript

JavaScript es un lenguaje de programación ligero, interpretado, o compilado con el método JIT (just-in-time). Es conocido con el lenguaje para páginas web, aunque también se usa en otros ambientes de desarrollo diferentes de los navegadores, como Node.js, Apache CouchDB y Adobe Acrobat. JavaScript es en lenguaje dinámico basado en prototipos que soporta varios paradigmas de programación, como la programación orientada a objetos, así como los estilos imperativo y declarativo. (Mozila Developer Network [MDN], 2019)

2.2.12. Git y GitHub

Git es un sistema de control de versiones (CVS por sus siglas en inglés) libre y de código abierto para manejar proyectos, desde pequeños hasta los de gran escala, con velocidad y eficiencia. Los CVS, registran los cambios hechos a archivos o conjunto de archivos en el tiempo, de modo que se puede acceder a versiones anteriores de ellos en un tiempo posterior. En el caso del desarrollo de software, los archivos controlados son principalmente de código fuente. (Git, 2019)

Github es una plataforma para el desarrollo colaborativo de software, que utiliza Git. Incluye herramientas para la revisión del código, la administración del proyecto, administración del equipo de trabajo, entre otras. Es uno de los almacenes de código más largo en el mundo, con alrededor de 100 millones de proyectos. Github tiene una comunidad de aproximadamente 36 millones de desarrolladores. (Github, 2019)

2.2.13. NodeJS y npm

Node es un entorno de ejecución de JavaScript orientado a eventos asíncronos, diseñado para construir aplicaciones en red escalables. Por su modelo de concurrencia basado en eventos asíncronos, node es más eficiente y fácil de usar que el modelo de concurrencia más común de hoy basado en hilos del sistema operativo. NodeJS usa por defecto npm para la gestión de paquetes. (Node.js Foundation, 2019)

NPM, es el registro de software más grande del mundo, que permite, que los desarrolladores de software de código abierto de cualquier parte del planeta, puedan compartir sus paquetes de software.

En un entorno de desarrollo con Node y npm, a través de la interface de línea de comandos (CLI por sus siglas en inglés) con que cuenta npm, los desarrolladores pueden administrar (instalar, remover y actualizar, etc.) los paquetes incorporados en sus aplicaciones. Npm dispone también de una página web para la búsqueda de paquetes y creación de perfiles para nuevos desarrolladores que quieran ser parte de la comunidad npm.

Aproximadamente, 1 millón de paquetes se encuentran registrados en npm, y 11,7 millones de descargas se hacen semanalmente. Muchos de las librerías escritas en JavaScript más populares se encuentra registradas en npm, tales como, openlayers. (NPM, 2019)

2.2.14. React

Es una librería para construir interfaces de usuario (UI por sus siglas en inglés) desarrollada por Facebook. Usa un estilo declarativo que hace sencilla la creación de UIs interactivas. El desarrollo de UIs se basa en la creación de componentes que encapsulan la apariencia y/o comportamiento de una porción de una UI y que luego se combinan para crear UIs más complejas. React es multiplataforma pues también puede ser usando en el servidor, así como para aplicaciones móviles. React está disponible en npm. (Facebook Inc., 2019)

2.2.15. Babel

Babel es un compilador de JavaScript, principalmente usado para convertir código de las últimas versiones del estándar ECMAScript (JavaScript) en código que es compatible tanto en navegadores actuales como antiguos. Además, es capaz de suplir características ausentes en algunas versiones de navegadores. (Babel, 2019)

2.2.16. UML

El lenguaje de modelado unificado (UML por sus siglas en inglés), actualmente en su revisión 2 (UML 2), tiene el objetivo de proveer a los arquitectos de sistemas, ingenieros de software, y desarrolladores de software de herramientas para el análisis, diseño, e implementación de sistemas basados en software, así como, para el modelamiento de negocios y procesos similares.

Las versiones iniciales de UML (UML 1) se originaron basadas en tres métodos líderes orientados a objetos (Booch, OMT and OOSE), e incorporaba buenas prácticas del diseño de lenguajes de modelado, programación orientada a objetos y lenguajes de descripción de arquitectura. En relación con UML 1, UML 2 ha sido mejorada con definiciones significativamente más precisas de su semántica y reglas de sintaxis abstractas, una estructura del lenguaje más modular, y una enormemente mejorada capacidad para el modelamiento de sistemas de gran escala. (Object Managment Group, 2019)

2.3.GLOSARIO DE TÉRMINOS

- **Sistema de coordenadas de referencia:** Sistema de coordenadas que permite la definición de un punto de la tierra por medio números (coordenadas).
- **Servidor de mapas:** Servidor con la capacidad de crear mapas.
- **Servidor WMS:** Servidor de mapas que adopta el estándar WMS como modelo para la creación de mapas.
- **ECQL:** Lenguaje parecido a SQL que en el contexto de los servicios WMS, WFS y WPS, sirve para definir condiciones para el filtrado de features.
- **Parámetros del fabricante:** Parámetros que no son parte de los estándares WMS, WFS y WPS, pero que son agregados para proveer capacidades mejoradas.
- **Bounding box:** Para un número finito de features, es la caja que los contiene, definida por dos coordenadas que señalan sus esquinas opuestas.

CAPÍTULO III. MARCO METODOLÓGICO

3.1.ENFOQUE Y DISEÑO

3.1.1. Enfoque

El presente trabajo de investigación se abordó usando el enfoque cuantitativo, pues este (el trabajo de investigación), se basa en la teoría y técnica existente para la creación de un producto que luego es objeto de evaluación, aplicando, para la recolección de datos, técnicas e instrumentos que se fundamentan en la medición. Existe además un carácter probatorio inherente al trabajo de investigación, en cuánto pone de manifiesto que la teoría y tecnología existente son perfectamente aplicables al desarrollo de un producto.

El enfoque cuantitativo se define como: Una aproximación de la investigación que es secuencial y probatoria. Cada etapa precede a la siguiente y no se pueden saltar o eludir pasos. La recolección de datos se fundamenta en la medición y se deben analizar con métodos estadísticos. La investigación cuantitativa debe ser lo más objetiva posible y debe desarrollarse sobre la realidad objetiva, externa e independiente de las creencias que tengamos sobre ella. Se aplica la lógica deductiva. De lo general a lo particular. (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014)

3.1.2. Diseño

El presente trabajo de investigación es no experimental – transversal, dado que no existe manipulación deliberada de variables y la recolección de datos se hace en un solo momento. Después de haber concluido con el desarrollo de la librería, esta se somete a un control de calidad y a una evaluación por parte de especialistas.

Se define la investigación no experimental como: La que se realiza sin manipular deliberadamente variables; donde se observan los fenómenos tal como se dan en su contexto natural, para analizarlos. (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014)

Además, se dice en relación a la investigación transeccional o transversal que: Los datos se recolectan en un solo momento. Su propósito, es describir variables y analizar su incidencia e interrelación en un momento dado. Es como "tomar una fotografía" de algo que sucede. (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014)

3.2.SUJETOS DE LA INVESTIGACIÓN

Debido a la poca disponibilidad de profesionales especialistas o con experiencia en el desarrollo de aplicaciones web de mapas, se consideró también la participación de profesionales de la carrera de Ingeniería Informática con conocimientos y experiencia laboral en el desarrollo de aplicaciones web, a quienes previamente se capacitó a fin de que estuviesen aptos para evaluar la librería. El temario de la capacitación se detalla en el documento del ANEXO 9.

Nombre completo	Profesión	Especialidad	Observación
Fabián Ary Merino Livia	Ingeniero Informático	Desarrollador web	Con experiencia en desarrollo de aplicaciones web de mapas y Openlayers.
Luis Gabriel Berrú Aguilar	Ingeniero Informático	Desarrollador web	Recibió capacitación.
Ángel Andrés Acosta Pingo	Ingeniero Informático	Desarrollador web	Recibió capacitación.

Tabla 12. Evaluadores de la librería.

Elaboración propia

3.3.MÉTODOS Y PROCEDIMIENTOS

3.3.1. Metodología de trabajo

El presente trabajo de investigación, se desarrolló siguiendo la metodología que se ilustra y describe a continuación:

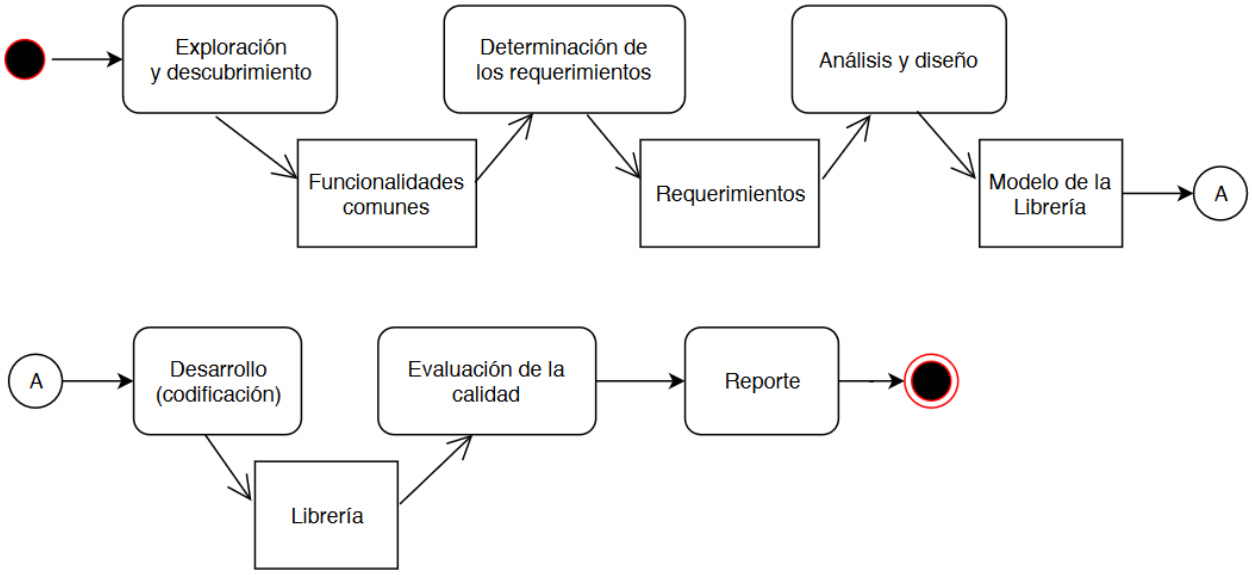


Figura 8. Diagrama de la metodología de trabajo.

Fuente: Elaboración propia

EXPLORACIÓN Y DESCUBRIMIENTO

En esta primera etapa, se visitaron los así denominados ‘visores de mapas’ que figuran en el sitio web de la Infraestructura de Datos Espaciales del Perú (IDEP); aplicativos mantenidos por entidades públicas y privadas, que permiten la exploración visual de información georreferenciada con un navegador web. (IDEP, 2019)

Esta visita se llevó a cabo para identificar funcionalidades que pudiesen ser incluidas en los requerimientos de la librería. Se visitaron todos los visores de mapas y se realizaron anotaciones sobre las funcionalidades que tenían y las características particulares de estas.

El resultado de esta exploración, fue una lista de funcionalidades comúnes que sirvieron como base para la determinación de requerimientos que complementarían los requerimientos de la librería.

DETERMINACIÓN DE LOS REQUERIMIENTOS

Se determinaron y describieron los requerimientos (funcionales y no funcionales) de la librería, haciendo énfasis en su relación con Openlayers y los servicios WMS, WFS y WPS de Geoserver.

Además, se identificaron, de las funcionalidades comúnes obtenidas de la etapa de 'Exploración y Descubrimiento', requerimientos que complementaron la lista de requerimientos de la librería.

El resultado de esta etapa, fue la especificación de los requerimientos del software.

ANÁLISIS Y DISEÑO

En esta etapa, se llevó a cabo un análisis y diseño orientado a objetos de la librería, usando el estándar UML. Se elaboraron una serie de diagramas que incluyen: diagrama de casos de uso, diagrama de clases, etc., para definir el sistema.

El resultado, fue un modelo de la librería que cumple con los requerimientos determinados en la etapa anterior.

DESARROLLO DE LA LIBRERÍA

En esta etapa, se llevó a cabo el desarrollo (codificación) de la librería, es decir, del modelo obtenido de la etapa anterior. La documentación para su uso se elaboró de manera conjunta.

EVALUACIÓN DE LA CALIDAD

En esta etapa, la librería se sometió a evaluación usando un método basado en el estándar ISO/IEC 25000:2005 (Software Engineering – Software product Quality Requirementes and Evaluation [SQUARE]). Las pruebas de calidad se realizaron para evaluar el cumplimiento de los requerimientos, así como la usabilidad de la librería.

REPORTE

Se presentaron los resultados de las evaluaciones, al igual que las conclusiones a las que se llegaron y recomendaciones en torno a la investigación y la librería.

3.3.2. Metodología de evaluación

La metodología de evaluación que se utilizó, se basó en el método de Baldeón Villanes (2015) para la evaluación de calidad del software basado en el estándar ISO/IEC 25000. Este método puede ser utilizado para diferentes etapas del ciclo de vida del software, la cual, para fin de su aplicación, se divide en cinco según el modelo de desarrollo en cascada. En cada etapa se evalúa un entregable diferente.

En el presente trabajo de investigación, se sometió a evaluación el producto terminado. Es decir, que el método se aplicó en la última etapa del ciclo de vida del software.

Etapa del ciclo de vida del software	Entregable	Descripción	Objetivo de la evaluación
Análisis	**Especificación de requerimientos del software**	Es el documento que contiene los requisitos funcionales y no funcionales documentados.	Verificar que los requisitos funcionales cubren las necesidades de los usuarios y stakeholders.
Diseño	**Diseño de la arquitectura del software**	Es el diagrama de la arquitectura del software. Los componentes del diagrama deben estar descritos.	Verificar que el diseño de la arquitectura sea soportado por la arquitectura existente de la empresa; es decir, esta evaluación debe verificar si la arquitectura diseñada es acorde a la arquitectura de la empresa, tanto en hardware, software, redes y conectividad y estándares de diseño.
Diseño	**Diseño de base de datos del software**	El diseño de la base de datos completa es el diagrama entidad relación y el diccionario de datos.	Verificar que el diseño de base de datos cumpla estándares de la empresa.
Implementación de código	**Código fuente del software**	Son los archivos que contienen el código fuente de la aplicación.	Verificar que el código fuente del producto software cumple los estándares de la empresa, así como los estándares de desarrollo seguro,

			mejores prácticas de desarrollo y las políticas de seguridad de información.
Implementación de software	**Software integrado**	Es el producto terminado publicado en el ambiente de pruebas.	Verificar que el software terminado cumple los requisitos funcionales y no funcionales definidos para el producto final.

Tabla 13. Entregables y objetivos del método de evaluación según las etapas del ciclo de vida del software.

Fuente: (Baldeón Villanes, 2015)

El método consta de cinco fases y está basado en los estándares ISO/IEC 25040 e ISO/IEC 25041. Las fases del método son:

- Fase 1: Establecer los requisitos de la evaluación
- Fase 2: Especificación de la evaluación
- Fase 3: Diseño de la evaluación
- Fase 4: Ejecución de la evaluación
- Fase 5: Conclusión de la evaluación

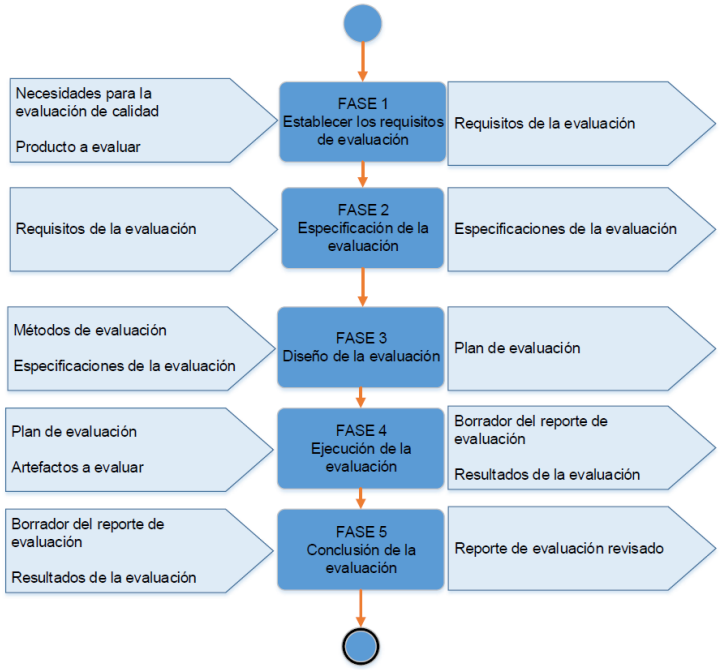


Figura 9. Fases del método de evaluación.

Fuente: (Baldeón Villanes, 2015)

FASE 1 – Establecer los requisitos de evaluación

El propósito de esta fase es definir los requisitos que se deben considerar en la evaluación. En esta fase se debe definir los componentes o productos que se evaluarán, así como, las características y sub-características de calidad (conforme al estándar ISO/IEC 25010) que se van a considerar en la evaluación. Los productos a evaluar corresponden con los entregables y se seleccionan según la etapa del ciclo de vida del software.

Esta fase del método se adecuó para que se puedan considerar en la evaluación, características particulares que el solicitante de la evaluación define y está interesado en evaluar, pero que no son parte de las características o sub-características de la calidad según el estándar ISO/IEC 25010.

El resultado de esta fase es el documento denominado '**Requisitos de Evaluación de Calidad**', que debe ser aprobado por el solicitante de la evaluación y el evaluador.

FASE 2 – Especificación de la evaluación

El propósito de esta fase es especificar las medidas de calidad, los métodos de evaluación a utilizar y los criterios de decisión para los requisitos definidos en la fase 1. En esta etapa se detalla la información necesaria para la evaluación por cada entregable susceptible a ser evaluado. Se debe considerar únicamente a los entregables incluidos en la evaluación. En el caso del entregable de la última etapa del ciclo de vida del software, la información necesaria es (1) los *Requisitos funcionales y no funcionales del sistema* y (2) el *software integrado y publicado*.

El resultado de esta fase es el documento denominado '**Especificaciones de la Evaluación de Calidad**', que debe ser aprobado por el solicitante y el evaluador.

FASE 3 – Diseño de la evaluación

El propósito de esta fase es planificar actividades de la evaluación de calidad. Para este proceso se debe tener en cuenta la disponibilidad de los recursos (personas, equipos, herramientas).

El resultado es un documento denominado '**Diseño de la Evaluación de Calidad**', que contiene principalmente el plan de trabajo de la evaluación de calidad. Este documento debe ser aprobado por el solicitante y el evaluador.

FASE 4 – Ejecución de la evaluación

El propósito de esta fase es ejecutar las actividades de la evaluación de calidad y documentarlas debidamente. Según el plan definido, el solicitante debe entregar los productos a evaluar, y el evaluador

debe registrar estos productos o sus componentes y los documentos relacionados. La información de registro debe contener los siguientes datos: nombre del componente, condición del componente (anomalías especiales o condiciones físicas), versión y fecha de recepción.

Producto de las acciones de evaluación se obtienen datos para producir los resultados que serán incluidos en el reporte de evaluación, estos datos pueden ser números, gráficos, diagramas entre otros. En resumen, el evaluador debe registrar todos los datos intermedios necesarios para los resultados del reporte de evaluación.

Si una herramienta de software es necesaria para recolectar datos o interpretarlos, una referencia a esta herramienta debe ser incluida en el reporte de evaluación. La información mínima será: el nombre de la herramienta, el fabricante y la versión.

Con la finalidad de lograr la máxima objetividad de la evaluación, es importante que todas las acciones de evaluación sean revisadas por una persona diferente a la que realizó la acción. Los resultados de esta revisión deben estar incluidos en los registros de la evaluación.

El resultado de esta fase es un borrador del reporte de evaluación de calidad.

FASE 5 – Conclusión de la evaluación

El propósito de esta fase es revisar y presentar el informe final de la evaluación de calidad. Es necesario evaluar la personalización del borrador del reporte de evaluación previo a su revisión con el público objetivo. Es decir, cuando el reporte va dirigido a personas con conocimientos técnicos, el nivel de detalle debe ser mayor; y cuando va dirigido a personas de nivel estratégico, la información debe tener un mayor nivel de abstracción. El resultado de esta fase, es el documento denominado '**Reporte de Evaluación de Calidad**'.

3.3.3. Exploración y Descubrimiento

Se visitaron 53 visores de mapas de un total de 89 que figuran en el sitio web de la IDEP. Los restantes, no pudieron ser visitados por motivos tales como: indisponibilidad del servicio por causas desconocidas; representaban una entrada duplicada en la lista; no eran visores de mapas; o fueron desarrollados con tecnologías no compatibles con las actualmente disponibles. Adicionalmente, se visitó el visor de mapas de la Empresa Prestadora de Servicios de Saneamiento GRAU (EPS GRAU).

Cabe recalcar sobre los visores de mapas visitados, que son mantenidos por las distintas entidades públicas y privadas del país, corresponden con diferentes áreas de estudio o interés, y varían en finalidad de uso y grados de complejidad; otorgándole así, diversidad al conjunto.

Clase	Visitados	No Visitados	Total
Sociedad	15	11	26
Transporte	3	6	9
Agricultura	2	1	3
Atmósfera, Climatología, Meteorología	2	0	2
Medio Ambiente	11	3	14
Localización	2	2	4
Planeamiento Catastral	1	3	4
Océanos	1	1	2
Redes de Suministro	9	4	13
Información Geo-científica	3	2	5
Aguas Interiores	3	2	5
Cobertura de la Tierra con Mapas Básicos e Imágenes	1	1	2
Salud	1	0	1
Total	**54**	**36**	**90**

Tabla 14. Número de visores de mapas según clasificación de la IDEP. (Incluyendo el visor de la EPS GRAU dentro de la clase 'Redes de suministro')

Fuente: (IDEP, 2019)

Elaboración propia

3.3.3.1. Funcionalidades comunes

De las funcionalidades comunes identificadas entre los visores de mapas visitados, cabe destacar las siguientes:

- Visualización de capas ráster.
- Visualización de capas vectoriales.
- Visualización de leyenda de las capas.
- Cambio de estilo de las capas.
- Filtro de elementos de las capas.
- Identificación de un elemento de una ubicación dada.
- Identificación de los elementos de un área específica.
- Búsqueda de elementos en las capas.
- Exportación de la información de los elementos del mapa.
- Generación de reportes basados en la información de los elementos del mapa.
- Dibujo de figuras geométricas.
- Medición.
- Impresión.
- Carga de datos espaciales.

A continuación, se describen cada una de las funcionalidades.

VISUALIZACIÓN DE CAPAS RÁSTER

De las características que se observaron de esta funcionalidad, cabe destacar las siguientes:

- La creación de las capas se basa en la realización de peticiones a un servicio web para obtener una imagen o conjunto de imágenes del área de interés, con las cuales se crea la capa.
- En algunos visores las capas se crean con ayuda de un servidor WMS (capas WMS), que provee las imágenes de las capas, mientras que, para otros, la naturaleza del servicio no se pudo determinar.

- En todos los visores, las capas se crean de un modo similar a aquellas que consultan a un servicio WMS. Todos siguen uno de los dos métodos que se identificaron y describen a continuación:

    a) Realizar una única petición al servidor de mapas para obtener una imagen de la capa que abarque el área de interés.

    b) Realizar varias peticiones asíncronas al servidor de mapas, para obtener un conjunto de imágenes de menor e igual tamaño que corresponden a porciones del área de interés.

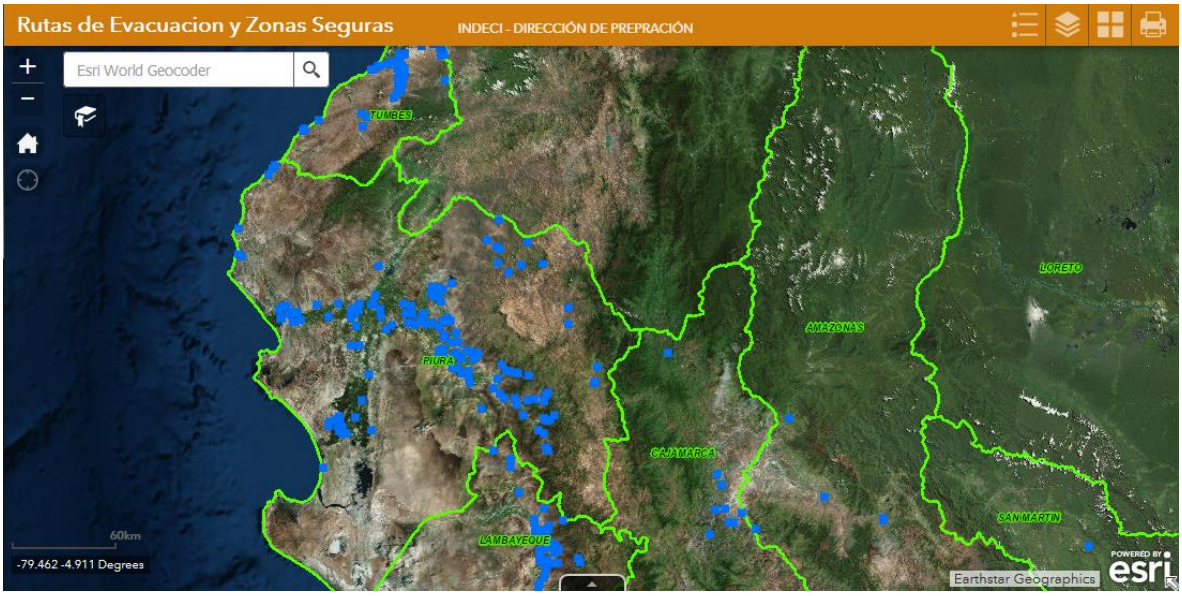


Figura 10. Visor de Rutas de Evacuación y Zonas Seguras del Instituto Nacional de Defensa Civil (INDECI), donde se pueden apreciar algunas capas ráster superpuestas.

Fuente: (INDECI, 2019)

Elaboración propia

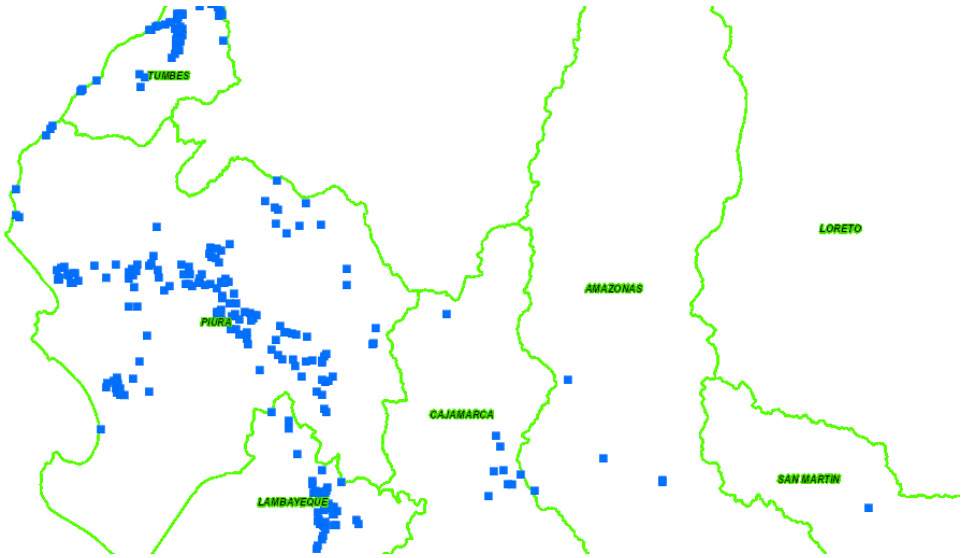


Figura 11. Imagen retornada por servicio WMS, usada para la creación de una capa ráster.

Fuente: (INDECI, 2019)

Elaboración propia.

VISUALIZACIÓN DE CAPAS VECTORIALES

De las características que se observaron de esta funcionalidad, cabe destacar las siguientes:

- A diferencia de las capas ráster, las capas vectoriales se crean a partir de la información de sus elementos y no de imágenes del área (nótese la relación que existe con los modelos de representación ráster y vectorial).
- Los visores obtienen la información de los elementos ejecutando una petición al servidor.

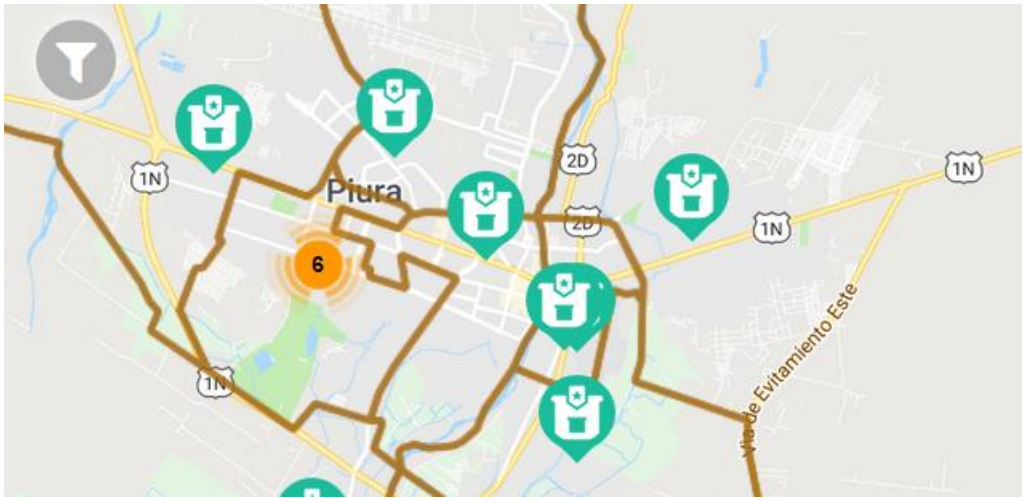


Figura 12. Aplicación web 'Ubica tu comisaría' del Ministerio del Interior (MININTER), que muestra una capa de tipo vectorial, de las comisarías del Perú.

Fuente: (MININTER, 2019)

Elaboración propia

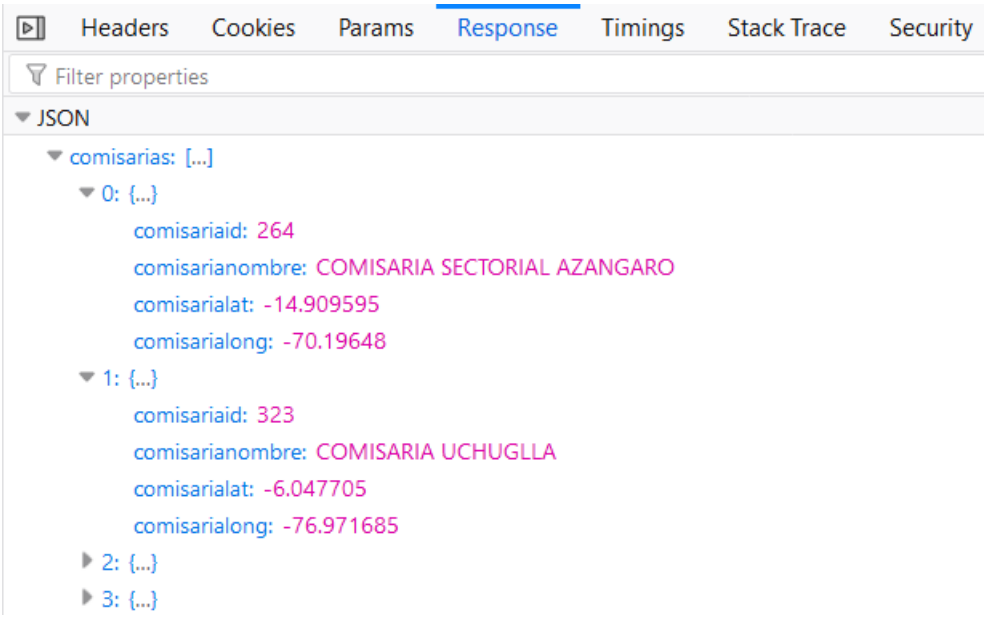


Figura 13. Respuesta de la petición que la aplicación 'Ubica tu comisaría', realiza para obtener la información de todas las comisarías del país y crear la capa de comisarías.

Fuente: (MININTER, 2019)

Elaboración propia

VISUALIZACIÓN DE LEYENDA DE LAS CAPAS

Se identificaron dos enfoques usados por los visores de mapas para crear una leyenda.

- **Creación en el cliente**. La leyenda es creada en el lado del cliente, con las imágenes de los símbolos de los elementos de las capas.
- **Creación en el servidor**. Se realiza una petición a un servicio WMS, que retorna la leyenda en la forma de una única imagen, en la que figuran, todos los símbolos de los elementos de las capas junto con su descripción.

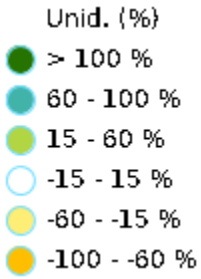


Figura 14. Leyenda obtenida de servicio WMS.

Fuente: (SENAMHI, 2019)

Elaboración propia

CAMBIO DE ESTILO DE LAS CAPAS

El visor de mapas de la EPS GRAU, contaba con una interface de usuario para cambiar el estilo de las capas. El usuario, puede elegir un estilo de la lista de estilos disponibles, que desea se aplique a la capa.

De las características que se observaron, cabe destacar la siguiente:

- Que esta funcionalidad está asociada a capas ráster creadas con un servicio WMS, pues el cambio de estilo es posible gracias al parámetro STYLES de la operación GetMap.

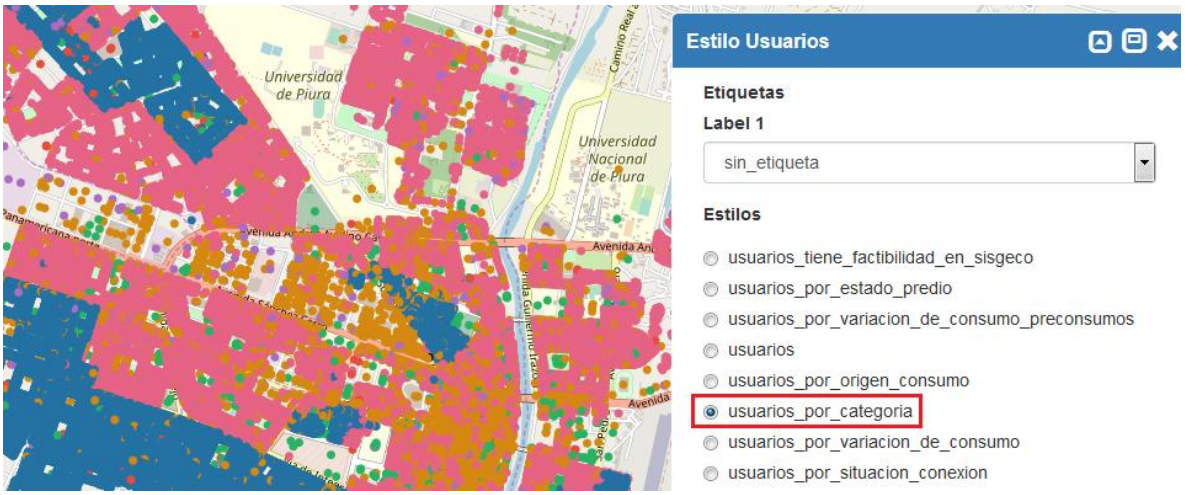


Figura 15. Capa 'usuarios' del visor de la EPS GRAU, junto con sus estilos.

Fuente: (EPS GRAU, 2019)

Elaboración propia

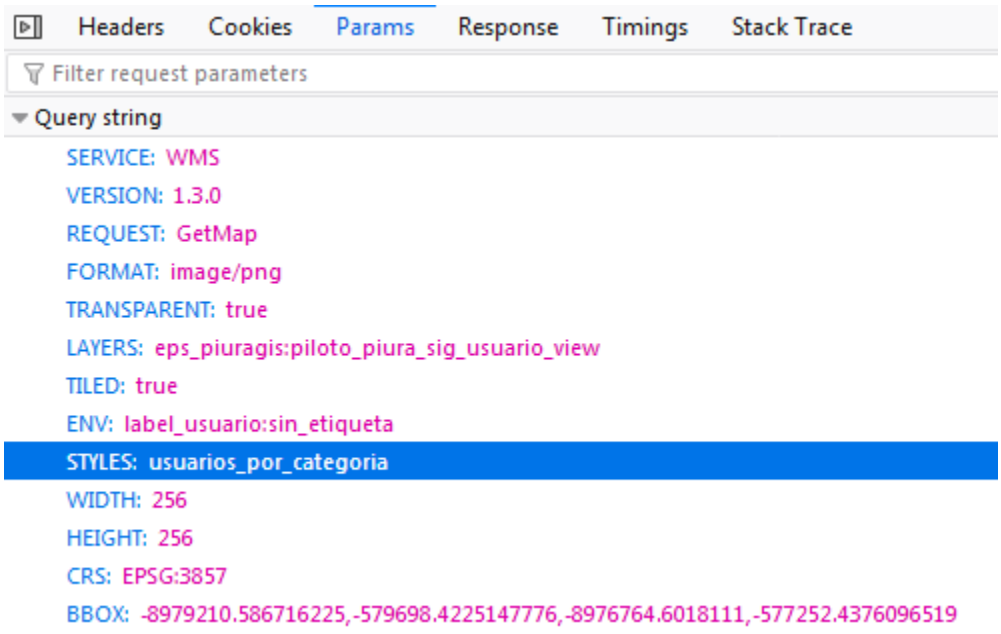


Figura 16. Petición GetMap usado para crear la capa 'usuarios' con un estilo específico. Note la presencia del parámetro 'STYLES' que define el estilo a usar.

Fuente: (EPS GRAU, 2019)

Elaboración propia

FILTRO DE ELEMENTOS DE LAS CAPAS

En algunos visores, se observó que se podía filtrar los elementos de las capas. Por ejemplo, en el visor de la EPS GRAU, que cuenta con una GUI para filtrar las tuberías de la red de agua potable, o en la aplicación SUSALUDmap, que permite filtrar los establecimientos de salud.

De las características que se observaron, cabe destacar las siguientes:

- En el caso de capas WMS, el filtro es posible gracias al parámetro CQL_FILTER de la operación GetMap.
- En el caso de capas vectoriales, el visor realiza una petición HTTP para obtener solo la información de los elementos de interés. La capa se actualiza entonces, descartando los elementos anteriores.

Figura 17. Capa 'tuberías' del visor de mapas de la EPS GRAU, filtrada según su tipo de material, para mostrar solo aquellas que están hechas de PVC.

Fuente: (EPS GRAU, 2019)

Elaboración

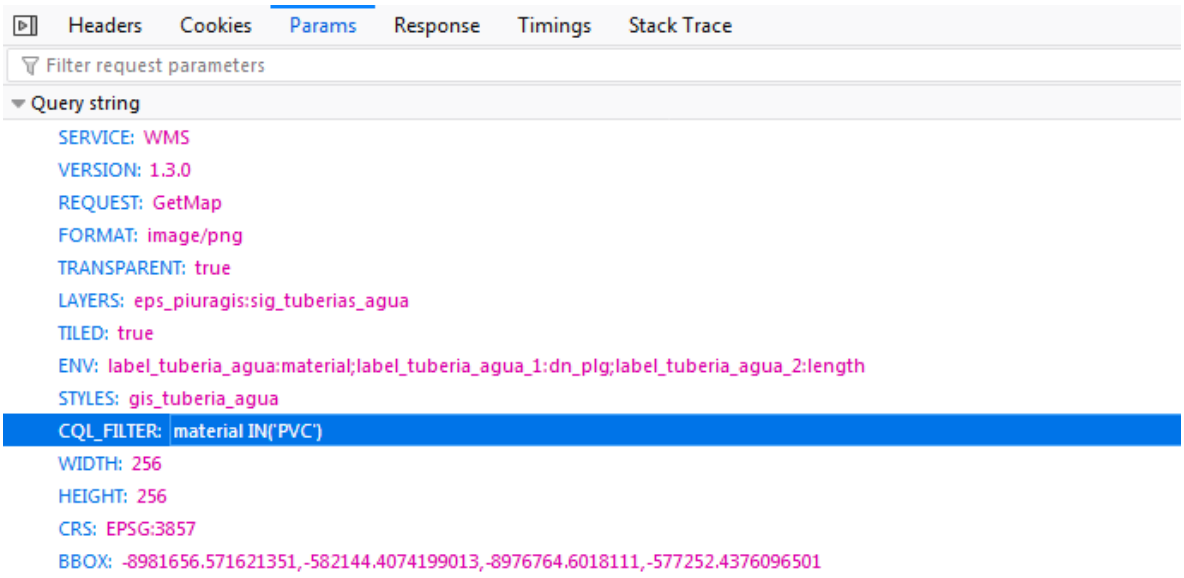


Figura 18. Petición GetMap usada para filtrar la capa 'Tubería'. Note la presencia del parámetro 'CQL_FILTER' que define el criterio utilizado para el filtro.

Fuente: (EPS GRAU, 2019)

Elaboración propia

IDENTIFICACIÓN DE UN ELEMENTO DE UNA UBICACIÓN DADA

Con esta funcionalidad, el usuario puede identificar los elementos de las capas. El usuario señala el elemento que quiere identificar dando clic sobre él en el mapa y el visor le muestra información de dicho elemento.

De las características que se observaron, cabe destacar las siguientes:

- Para identificar un elemento de una capa ráster, el visor realiza una petición HTTP para obtener los elementos que se encuentran en la ubicación señalada por el usuario. Dicha petición, incluye en sus parámetros, las coordenadas de la ubicación.
- En el caso de capas WMS, la identificación se logra invocando a la operación GetFeatureInfo.
- En el caso de capas vectoriales, no se realiza petición alguna, pues ya se cuenta con la información de los elementos (que se obtuvo para la creación de la capa).

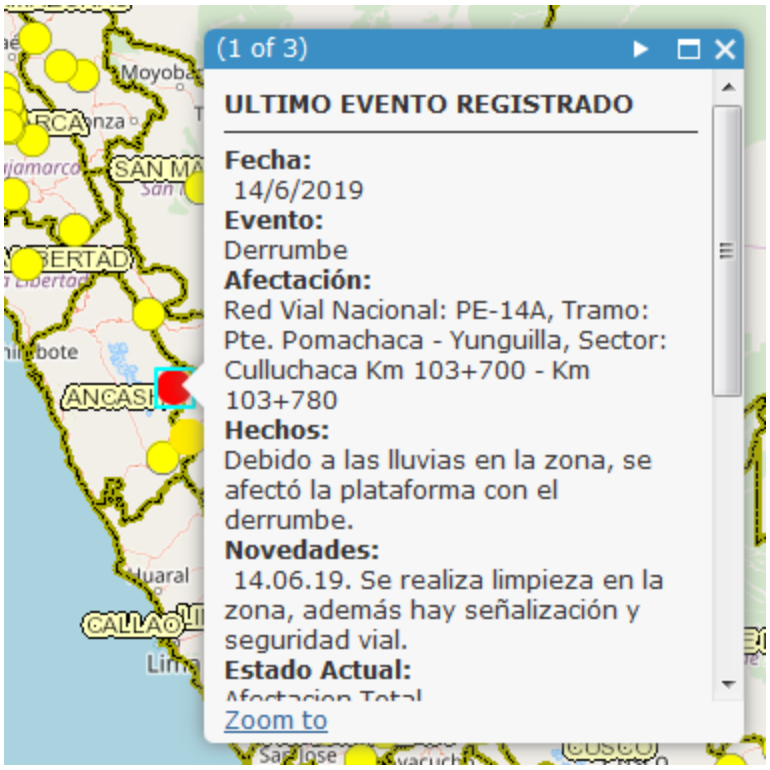


Figura 19. Identificación de una emergencia en el Visor de Emergencias Viales del Ministerio de Transporte y Comunicaciones (MTC).

Fuente: (MTC, 2019)

Elaboración propia

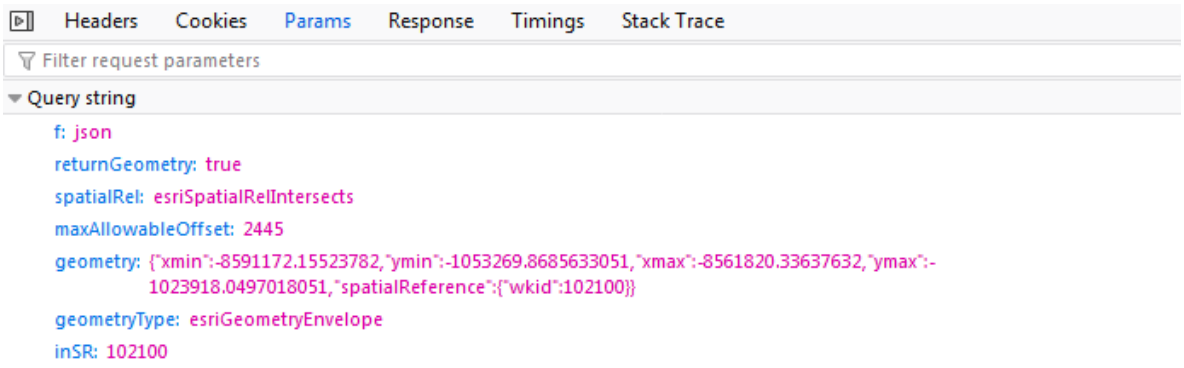


Figura 20. Petición HTTP para identificar emergencias en una ubicación específica. Note la presencia de una propiedad 'geometry' que señala la ubicación.

Fuente: (MTC, 2019)

Elaboración propia.

IDENTIFICACIÓN DE LOS ELEMENTOS DE UN ÁREA ESPECÍFICA

Esta funcionalidad permite la identificación de elementos de las capas que se encuentre dentro de un área específica. El usuario dibuja primero el área de interés sobre el mapa y el visor le muestra la información de todos los elementos que se hallen dentro de ella.

A diferencia de la identificación en una ubicación específica, esta solo se observó para capas ráster.

De las características que se observaron, cabe destacar las siguientes:

- Para identificar elementos de una capa ráster, el visor realiza una petición HTTP para obtener los elementos que se encuentran en el área señalada por el usuario. Dicha petición, incluye en sus parámetros, una descripción del área.
- En el caso de capas WMS, la identificación se logra invocando a la operación GetFeature de WFS.

Figura 21. Identificación de elementos en un área. Delimitación del área.

Fuente: (MINAM, 2019)

Elaboración propia

Figura 22. Identificación de elementos en un área. Resultados.

Fuente: (MINAM, 2019)

Elaboración propia

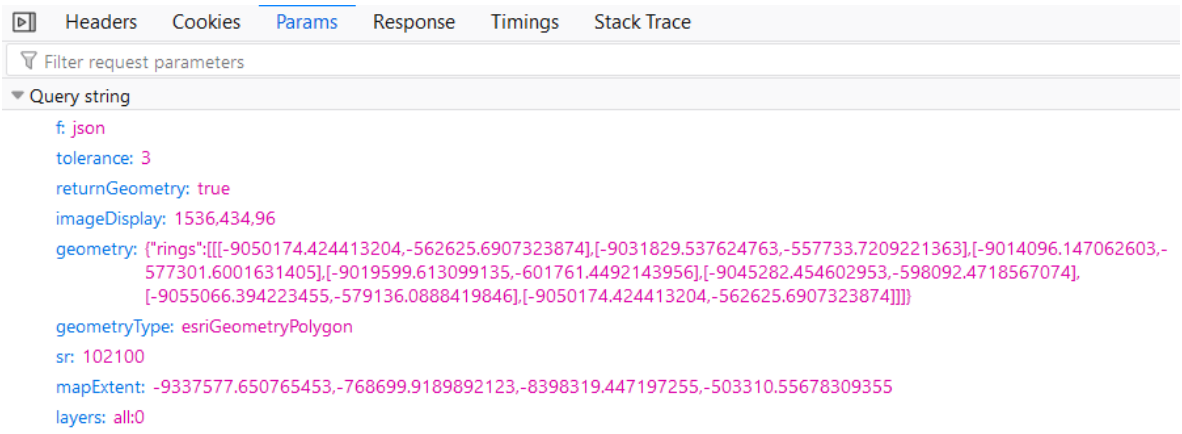


Figura 23. Petición HTTP para la identificación de elementos en un área específica. Note la presencia del parámetro 'geometry' que describe el área.

Fuente: (MINAM, 2019)

Elaboración propia

BÚSQUEDA DE ELEMENTOS EN LAS CAPAS

Los visores que incluyen esta funcionalidad, proporcionan al usuario una GUI que permite definir el criterio de búsqueda. Después de encontrar los elementos que cumplen con el criterio, estos se muestran/resaltan en el mapa.

De las características que se observaron, cabe destacar las siguientes:

- Para realizar la búsqueda, se realiza una petición HTTP con la que se obtienen los elementos de interés. El criterio de búsqueda se incluye en los parámetros de la petición.
- En las capas WMS, la búsqueda se lleva a cabo invocando la operación GetFeature de WFS.

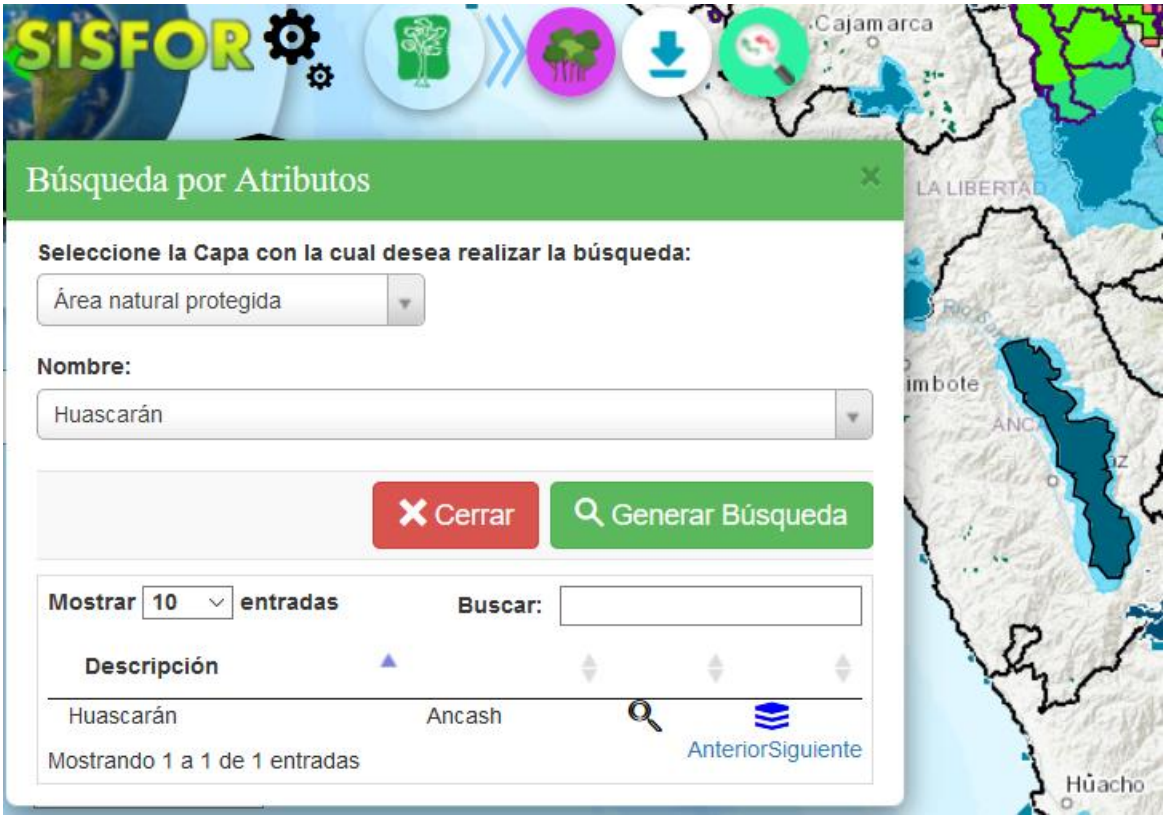


Figura 24. GUI para la búsqueda de elementos de las capas en el visor de mapas del Organismo de Supervisión de los Recursos Forestales y de Fauna Silvestre (OSINFOR).

Fuente: (OSINFOR, 2019)

Elaboración propia

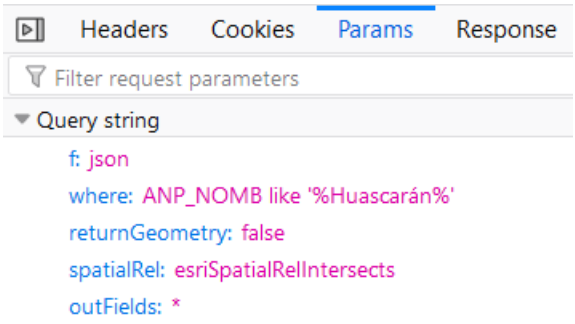


Figura 25. Petición HTTP para la búsqueda de elementos. Note la presencia del parámetro 'where' que señala el criterio de búsqueda.

Fuente: (OSINFOR, 2019)

Elaboración propia

EXPORTACIÓN DE LA INFORMACIÓN DE LOS ELEMENTOS DEL MAPA

Con esta funcionalidad, el usuario puede exportar la información de los elementos del mapa.

De las características que se observaron, cabe destacar las siguientes:

- La información se puede exportar en archivos con diferentes formatos, como: shapefile, CSV o DXF.
- Se puede restringir espacialmente la exportación. Es decir, se puede definir un área para limitar los elementos exportados a aquellos que se hallan dentro de ella.
- En algunos visores, se puede especificar el sistema de coordenadas de referencia en que se exportarán los elementos.

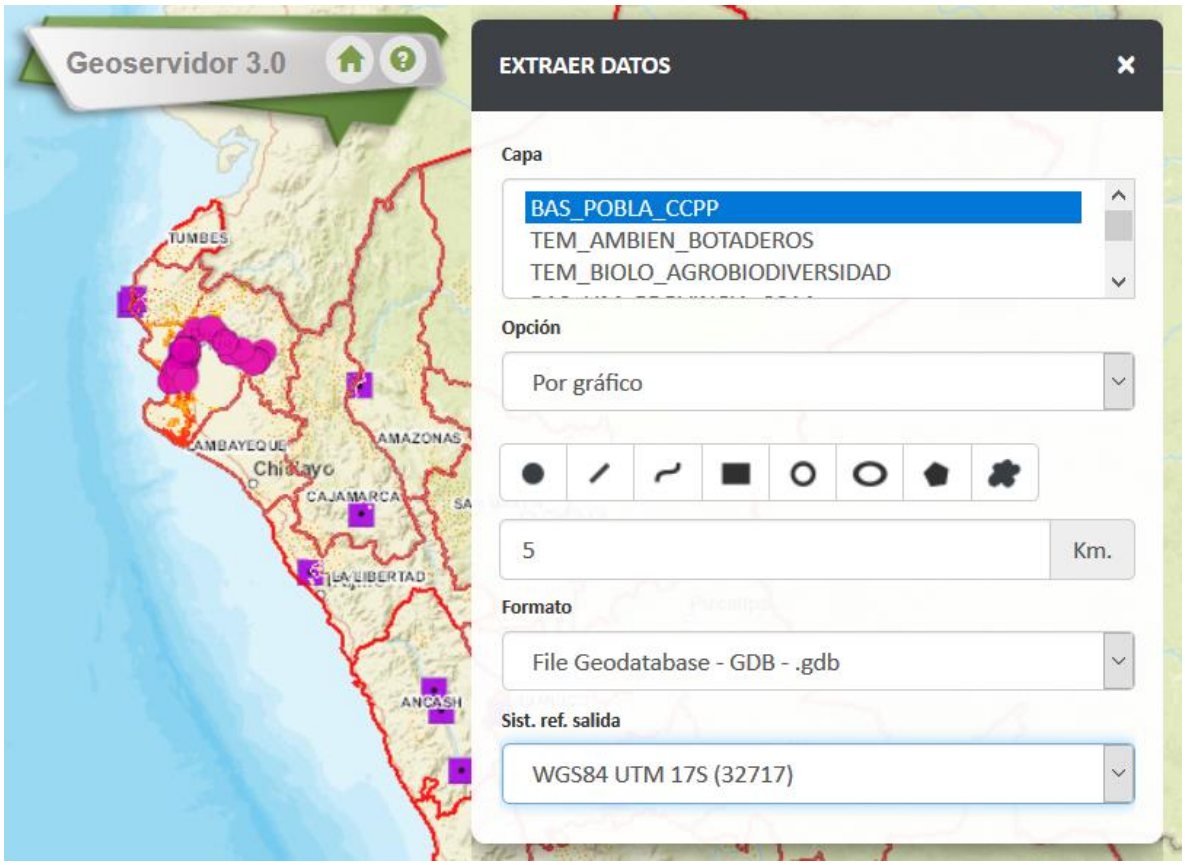


Figura 26. GUI para la exportación elementos de las capas.

Fuente: (MINAM, Geoservidor, 2019)

Elaboración propia

GENERACIÓN DE REPORTES BASADOS EN LA INFORMACIÓN DE LOS ELEMENTOS DEL MAPA

La información de los elementos de las capas ráster puede ser procesada para generar reportes. Tal como se observó en el visor de la EPS GRAU, que es capaz de emitir un reporte de la distancia total que existe en tuberías de agua potable por tipo de material.

De las características que se observaron, cabe mencionar las siguientes:

- Para generar el reporte, se invoca a la operación Execute de WPS, para que ejecute un proceso de agregación sobre los datos de la capa ‘tuberías’.
- Los datos del reporte siempre guardan correspondencia con el contenido de la capa. Es decir, que en caso de que la capa este filtrada, se aplica el proceso de agregación solo a los elementos que cumplen con el criterio de filtrado.

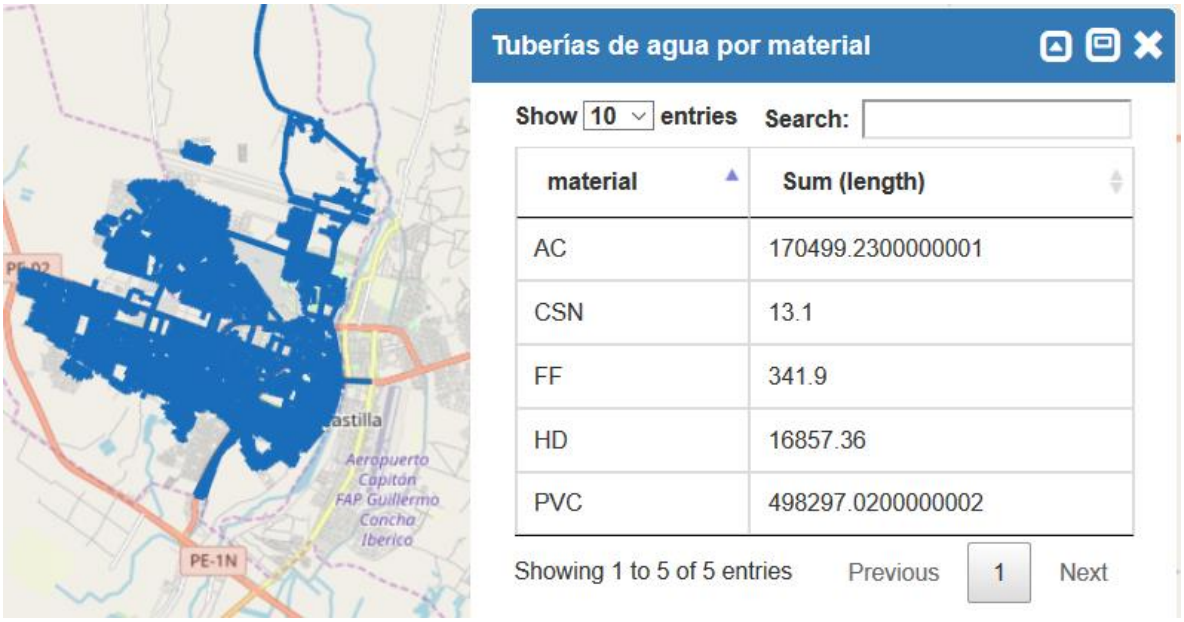


Figura 27. Reporte de la distancia total de tuberías de agua por material, del distrito de Piura.

Fuente: (EPS GRAU, 2019)

Elaboración propia.

DIBUJO DE FIGURAS GEOMÉTRICAS

De las características que se observaron de esta funcionalidad, cabe destacar las siguientes:

- El dibujo de figuras geométricas incluye otros tipos además de los básicos (punto, línea y polígono), como circunferencia, elipsoide e incluso texto.
- El aspecto del dibujo (estilo), puede ser definido por el usuario.
- El estilo de la figura podía cambiar una vez finalizado el dibujo.

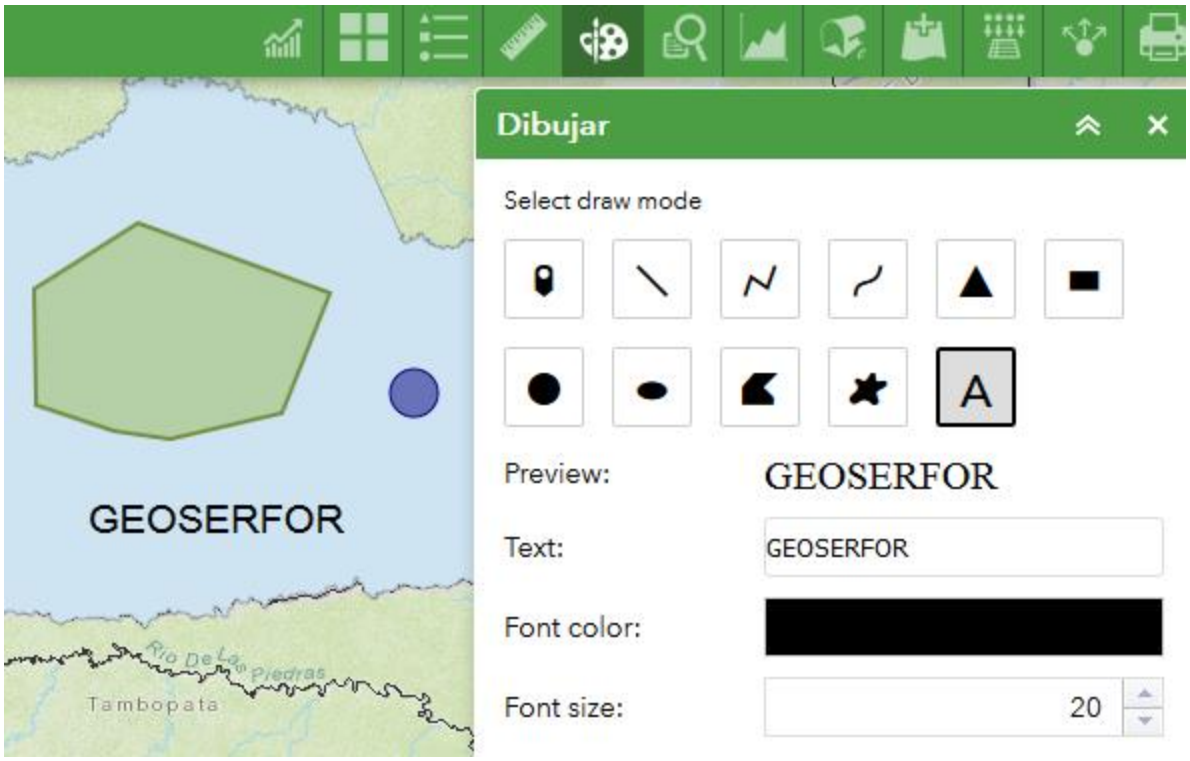


Figura 28. GUI para el dibujo de figuras geométricas en el visor del Servicio Nacional Forestal y de Fauna Silvestre (SERFOR).

Fuente: (SERFOR, 2019)

Elaboración propia

MEDICIÓN

De las características que se observaron de esta funcionalidad, cabe destacar las siguientes:

- Se apoya en la funcionalidad de dibujo para señalar las distancias o áreas a medir.
- El valor medido se puede mostrar una vez finalizado el dibujo, o incluso durante su ejecución.
- Se puede determinar la unidad en que será expresada el valor de la medición.

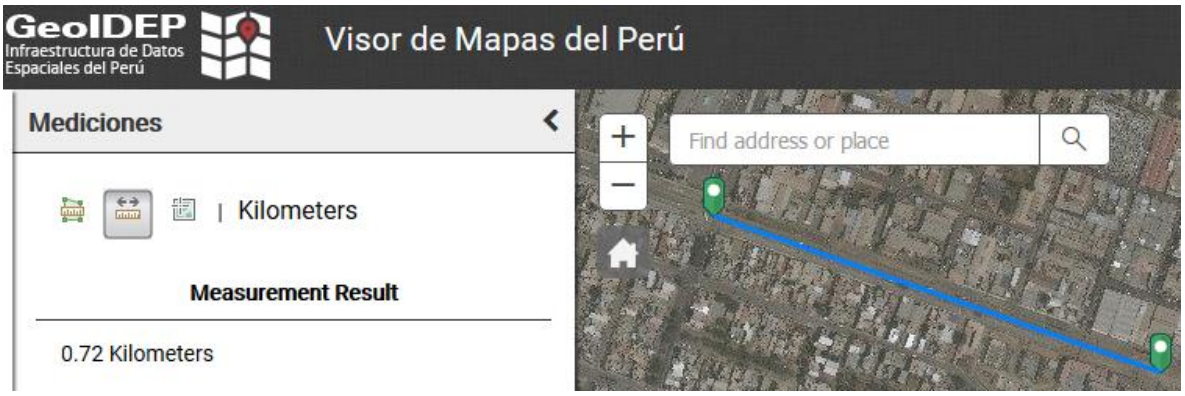


Figura 29. Medición en el Visor de Mapas del Perú de la IDEP.

Fuente: (IDEP, Visor de Mapas del Perú, 2019)

Elaboración propia

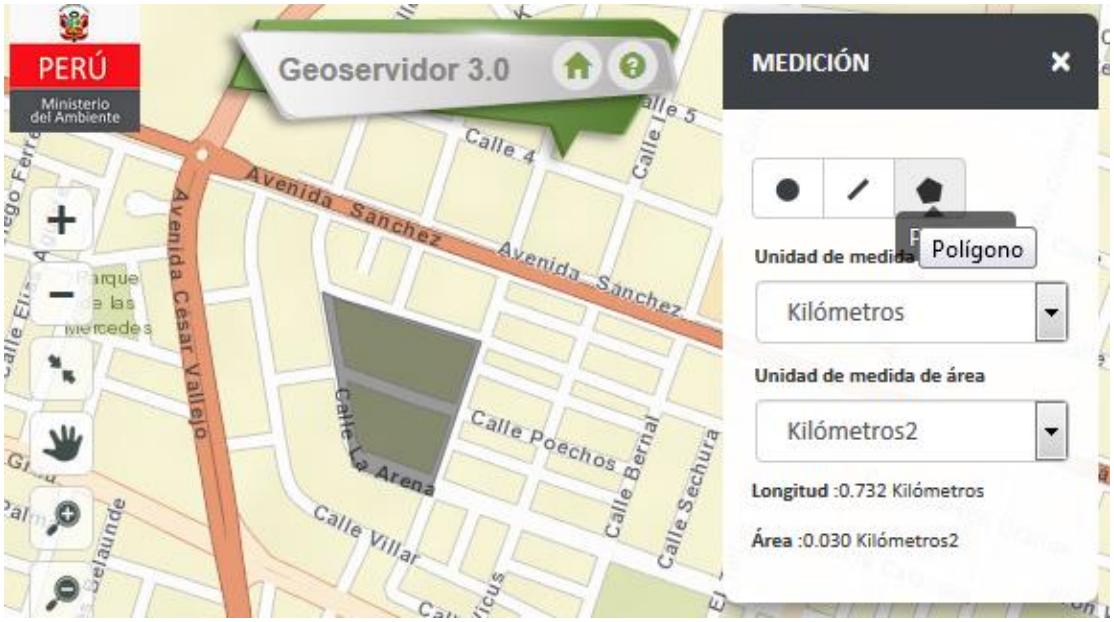


Figura 30. Medición en el Geoservidor del MINAM.

Fuente: (MINAM, Geoservidor, 2019)

Elaboración propia

IMPRESIÓN

Esta funcionalidad consiste en la generación de un archivo en algún formato de imagen, que contiene una captura del mapa junto con información relacionada.

De las características que se observaron, cabe destacar la siguiente:

- Se pueden definir los valores de algunos parámetros a considerar en la impresión, tales como:

    a) Escala.

    b) Formato de salida.

    c) Sistema de Coordenadas de Referencia a usar.

    d) Tamaño de la impresión. (A4, A3, etc.)

    e) Ancho y alto del mapa.

    f) Calidad de la imagen.

    g) Otros.

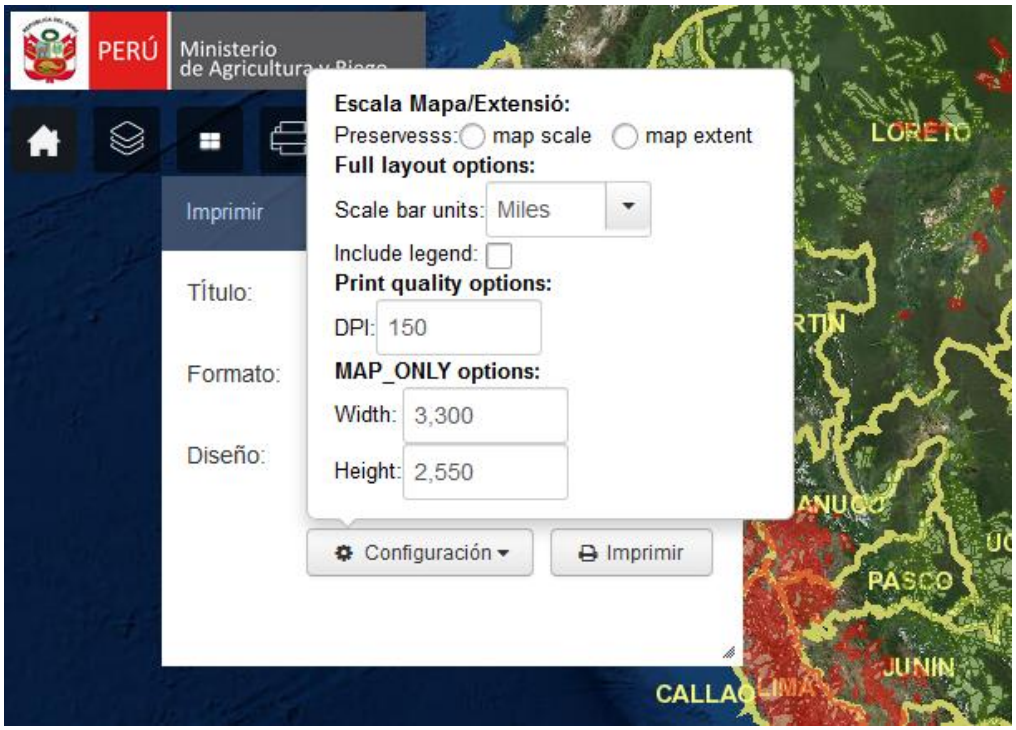


Figura 31. Formulario para la impresión de un mapa del visor de mapas del Ministerio de Agricultura y Riego (MINAGRI).

Fuente: (MINAGRI, 2019)

Elaboración propia

CARGA DE DATOS ESPACIALES

De las características que se observaron de esta funcionalidad, cabe destacar las siguientes:

- Los datos se pueden cargar de archivos con formatos tales como: shapefile, KML y CSV.
- Se pueden definir algunas características básicas del estilo que tendrán los elementos una vez sean cargados, por ejemplo: color de relleno y color de línea.
- La carga de datos espaciales supone la creación de una nueva capa vectorial.

Figura 32. GUI para subir archivo shapefile del visor mapas del OSINFOR.

Fuente: (OSINFOR, 2019)

Elaboración propia

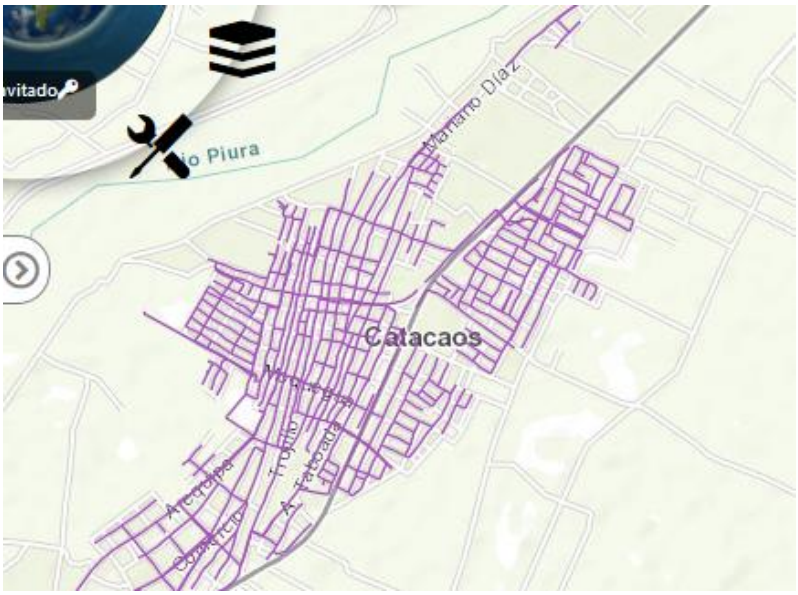


Figura 33. Archivo shapefile cargado en visor de mapas del OSINFOR.

Fuente: (OSINFOR, 2019)

Elaboración propia

3.3.4. Determinación de los Requerimientos

Los requerimientos se determinaron según los siguientes criterios:

- Debe incluir funcionalidades relacionadas a los servicios WMS, WFS y WPS de GeoServer.
- Sobre algunas funcionalidades comunes, debe simplificar su uso con respecto de la forma de uso de Openlayers.
- Debe ser compatible con Openlayers.

Nota: Para resumen y mejor entendimiento de la definición de los requerimientos se debe tener en cuenta que, cuando no se indique explícitamente, si se habla de capas, se estará haciendo alusión a capas WMS, y cuando se hable de elementos, a menos que se indique algo diferente, se estará haciendo referencia a elementos de las capas WMS. Adicionalmente, se usa el término feature para referirse a los elementos de las capas en un sentido más amplio, no solo elementos de capas, sino elementos y su información geográfica (geometría y propiedades). Openlayers cuenta con una clase para crear objetos que representan features, estos, son llamados features vectoriales.

3.3.4.1.Requerimientos funcionales

La librería, debe incluir funciones para cumplir con los siguientes requerimientos.

Requerimientos funcionales Funcionalidades relacionadas con los servicios WMS, WFS y WPS de GeoServer	
Funcionalidad	**Descripción**
Crear una capa WMS	Con la librería, se debe poder crear una capa WMS, constituida de teselas de imágenes que son enviadas por el servidor WMS.
Cambiar estilo de capa WMS	La librería, debe permitir, el cambio del estilo de una capa WMS. Esto es posible gracias al parámetro **STYLES** de la operación **GetMap - WMS**.
Filtrar contenido de capa WMS	Con la librería, debe ser posible filtrar el contenido de una capa WMS. Esto, gracias al parámetro **CQL_FILTER** de la operación **GetMap - WMS.**
Obtener información de los elementos en una ubicación dada	La librería debe ser capaz de retornar los features de una capa WMS en una ubicación dada, determinada por sus coordenadas geográfica. Se debe utilizar la operación **GetFeatureInfo –WMS**.
Obtener leyenda	Con la librería, se debe poder obtener la leyenda de una capa WMS en la forma de una imagen. Se debe utilizar la operación **GetLegendGraphic – WMS**.
Obtener información de los elementos	La librería debe ser capaz de retornar los features de una capa WMS. Se debe utilizar la operación **GetFeature – WFS**, de modo que se puedan aprovechar las capacidades de dicha operación, como por ejemplo, para delimitar a un área los features que se van a consultar.
Exportar información de los elementos	La librería, debe permitir la exportación de los features de una capa WMS en cualquiera de los formatos soportados por Geoserver. Se debe utilizar la operación **GetFeature – WFS**.
Obtener el número de elementos	Con la librería, se debe poder determinar el número de elementos que tiene una capa. Se debe utilizar la operación **GetFeature – WFS**.
Obtener descripción del tipo de elementos	Con la librería, se debe poder obtener una descripción del tipo de feature de una capa. Se debe utilizar la operación **DescribeFeatureType - WFS**.
Obtener el *bounding box* de una capa	Con la librería, se debe poder obtener el *bounding box* de una capa. Se debe utilizar la operación **Execute – WPS**.
Ejecutar funciones de *Agregación* a la	Con la librería, se debe poder ejecutar funciones de agregación sobre los features. Se debe utilizar la operación **Execute – WPS**.

información de los elementos	
Funcionalidades en común con Openlayers	
Crear un estilo para feature vectorial	Con la librería, se debe poder crear un estilo para un feature vectorial.
Dibujar feature vectorial	Con la librería, se debe poder dibujar features vectoriales.

Tabla 15. Requerimientos funcionales de la librería.

Fuente: Elaboración propia

3.3.4.2. Requerimientos no funcionales

Requerimientos no funcionales	
Característica	**Descripción**
Usabilidad	Para aquella funcionalidades en común con Openlayers, la librería debe ofrecer un forma de uso más sencilla.
Compatibilidad	La librería debe poder operar junto con Openlayers sin perjudicar su buen funcionamiento, ni viceversa.

Tabla 16. Requerimientos no funcionales de la librería.

Fuente: Elaboración propia

3.3.5. Análisis y Diseño

3.3.5.1. Módulos

Se hizo una división modular de la librería, que convenientemente se corresponde con los servicios WMS, WFS y WPS. Concretamente, el módulo OpenlayersLite incluye las funcionalidades que la librería tiene en común con Openlayers, que son una porción pequeña del total (de ahí el nombre), pero que proveen una forma de uso más sencilla.

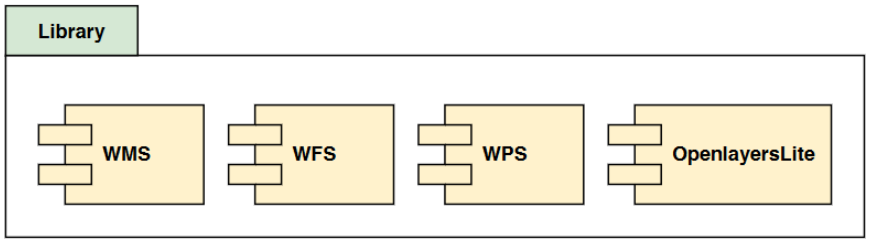


Figura 34. Módulos que constituyen la librería.

Fuente: Elaboración propia

3.3.5.2. Dependencias

Dado que la librería está basada en Openlayers, hace uso de algunos de sus elementos, los cuáles se describen en la siguiente tabla:

Elementos de Openlayers que usa la librería	
Map	Esta clase es el componente principal de Openlayers, representa y hace posible la integración de un mapa dinámico en un sitio web.
TileLayer	Clase para la creación de capas de fuentes que provee teselas de imágenes organizadas por niveles de zoom para resoluciones específicas (mismas que puede generar un servidor WMS).
TileWMS	Clase para la creación de fuentes que proveen imágenes divididas en teselas provenientes de servidores WMS.
Feature	Clase para la creación de objetos vectoriales para features, con su geometría y atributos, similar a los features en formatos de archivos vectoriales como GeoJSON.
VectorSource	Clase que representa una fuente de features para capas vectoriales.
GeoJSON	Clase para la conversión de features en formato JSON a objetos de la clase Feature de Openlayers, y viceversa.
ProjectionLike	Tipo de dato que representa un SRS. Su valor es el identificador del SRS en la forma

	de una cadena de caracteres.
Style	Clase que representa un contenedor de estilos para features.
Fill	Clase que sirve para definir el estilo de relleno para features.
Stroke	Clase que sirve para definir el estilo de borde para features.
CircleStyle	Clase que sirve para definir un estilo de círculo para features.
Icon	Clase que sirve para definir un estilo de ícono para features.
Text	Clase que sirve para definir el estilo de texto para features.

Tabla 17. Elementos de Openlayers que usa la librería.

Fuente: (Openlayers, 2019)

Elaboración propia

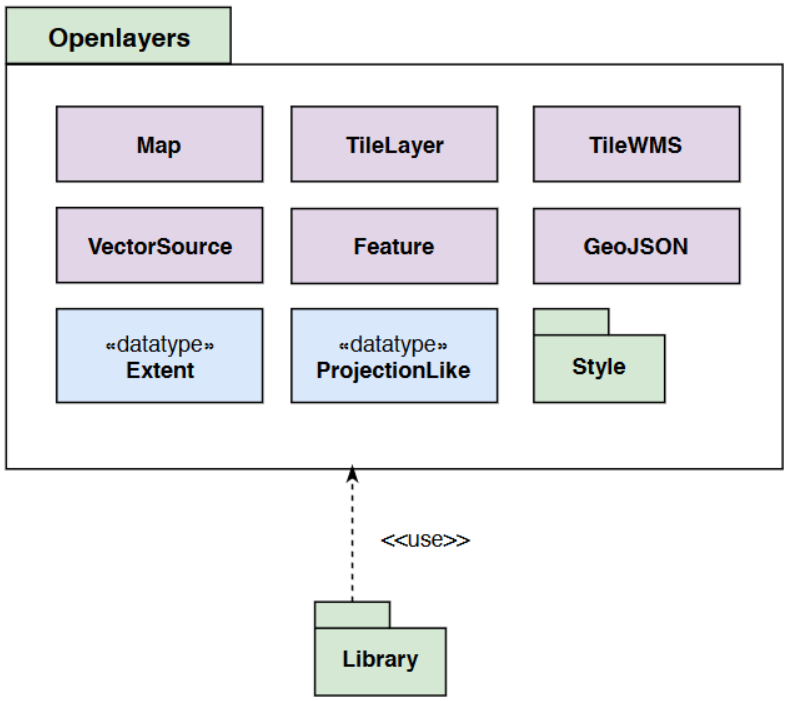


Figura 35. Dependencias de la librería.

Fuente: Elaboración propia

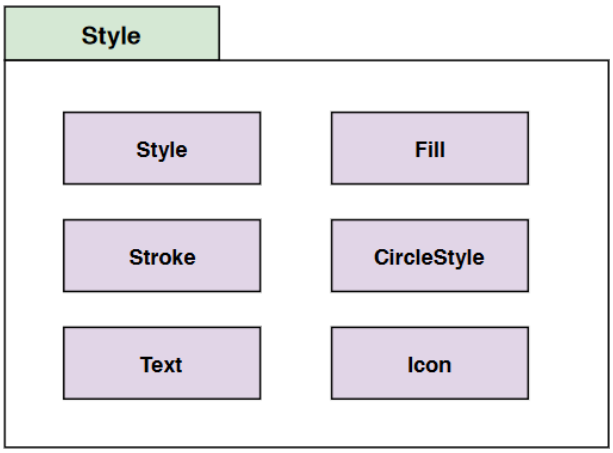


Figura 36. Paquete 'Style'.

Fuente: Elaboración propia

3.3.5.3. Módulos WMS

3.3.5.3.1. Diagrama de casos de uso

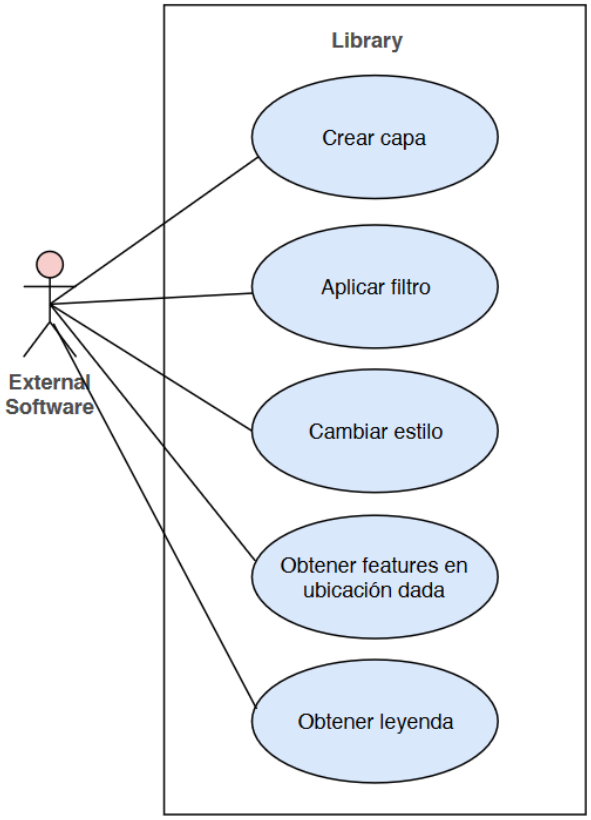


Figura 37. Diagrama de Casos de Uso del módulo WMS.

Fuente: Elaboración propia

A continuación, se detallan los casos de uso.

Detalle de caso de uso	
Nombre	Crear capa.
Actores	Software externo.
Función	Crear una capa WMS.
Descripción	El software externo puede crear un objeto que representa una capa WMS del mismo tipo que las que se crean con la clase *TileLayer*.

Tabla 18. Detalle del caso de uso 'Crear capa'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Aplicar filtro.
Actores	Software externo.
Función	Aplicar filtro a una capa WMS.
Descripción	El software externo puede aplicar a una capa WMS, un filtro. El resultado, será la capa que muestra solo los elementos que satisfacen la condición expresada por el filtro.

Tabla 19. Detalle del caso de uso 'Aplicar filtro'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Cambiar estilo.
Actores	Software externo.
Función	Cambiar el estilo de presentación de una capa WMS.
Descripción	El software externo puede cambiar el estilo de presentación de una capa WMS. El resultado, será la capa que es mostrada con el estilo elegido.

Tabla 20. Detalle del caso de uso 'Cambiar estilo'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Obtener features en una ubicación dada.
Actores	Software externo.
Función	Obtener features de una capa WMS en una posición dada.
Descripción	El software externo puede obtener los features de una capa WMS en una ubicación definida por sus coordenadas. El resultado es un array de objetos de la clase *Feature*.

Tabla 21. Detalle del caso de uso ‘Obtener features en una ubicación dado’.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Obtener leyenda
Actores	Software externo.
Función	Obtener la leyenda de una capa WMS.
Descripción	El software externo puede obtener la leyenda de una capa WMS. El resultado es una imagen.

Tabla 22. Detalle del caso de uso 'Obtener leyenda'.

Fuente: Elaboración propia

3.3.5.3.2. Tipos de datos

De los tipos de datos definidos, cabe mencionar sobre el tipo de dato GeoserverLayerOptions, que representa los parámetros más importantes en la creación de un objeto de la clase TileLayer. De los parámetros que se definen, cabe aclarar que: (1) el valor del parámetro *url* debe ser la URL del servicio WMS y (2) el valor del parámetro *crossOrigin* depende de si se quiere obtener la información de la capa que hay en un pixel (petición GetFeatureInfo), situación normada por la política samge-origin. (Openlayers, 2019)

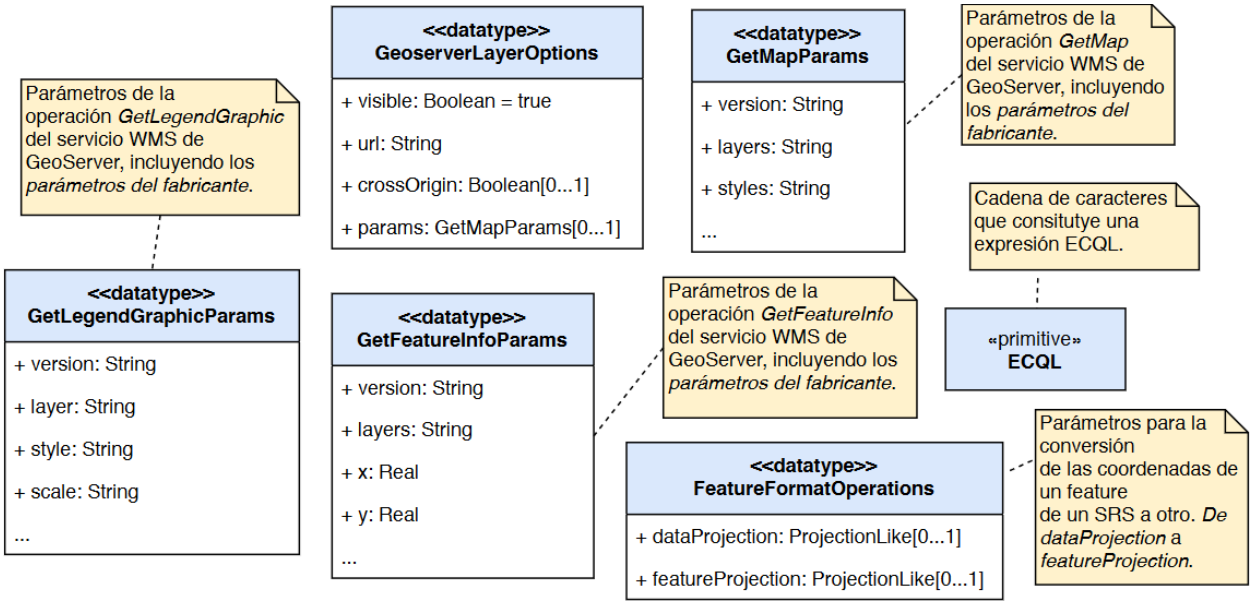


Figura 38. Tipos de datos del módulo WMS.

Fuente: Elaboración propia

3.3.5.3.3. Diagramas de clases

Se definieron tres interfaces que agrupan una serie de operaciones relacionadas a los servicios WMS, WFS y WPS de Geoserver. Estas interfaces son implementadas por la clase GeoserverLayer, que es una subclase de TileLayer.

Dada la relación de herencia entre las clases GeoserverLayer y TileLayer, la primera puede ser usada en lugar de TileLayer (más no viceversa) pues cuenta con las mismas características. Esta situación viene a propósito, pues suscita la integración con Openlayers.

En el diagrama UML que se muestra a continuación, se pueden observar las operaciones que define la interfaz WMS. El contenido de las demás interfaces se muestra más adelante en el desarrollo de los otros módulos.

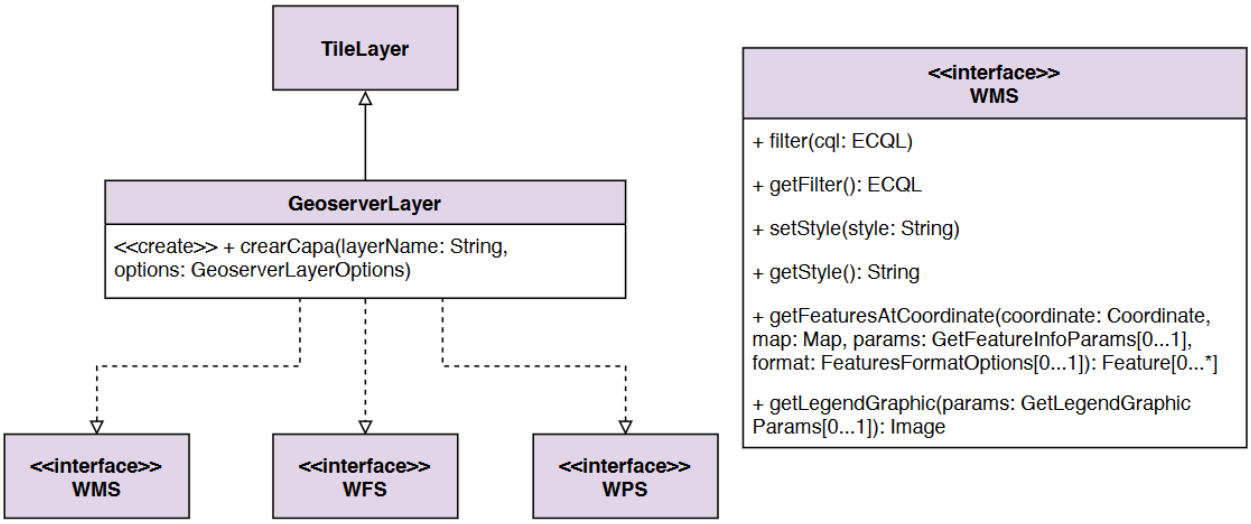


Figura 39. Diagrama de Clases del módulo WMS.

Fuente: Elaboración propia

3.3.5.3.4. Diagramas de interacción

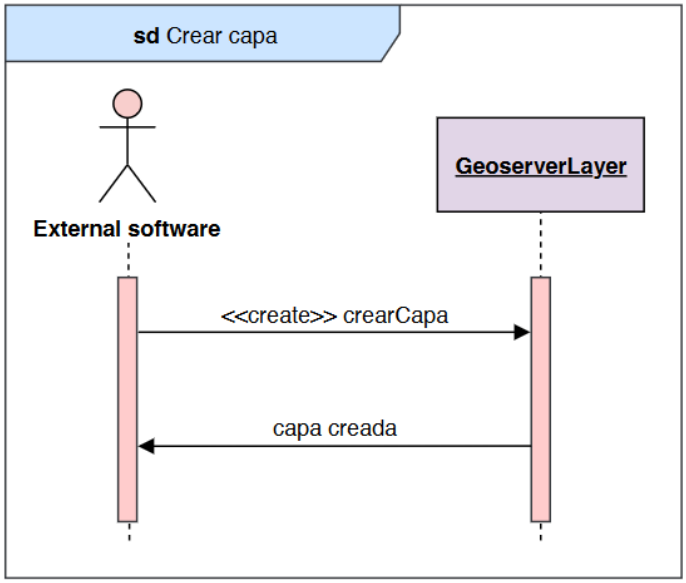


Figura 40. Diagrama de interacción para crear una capa.

Fuente: Elaboración propia

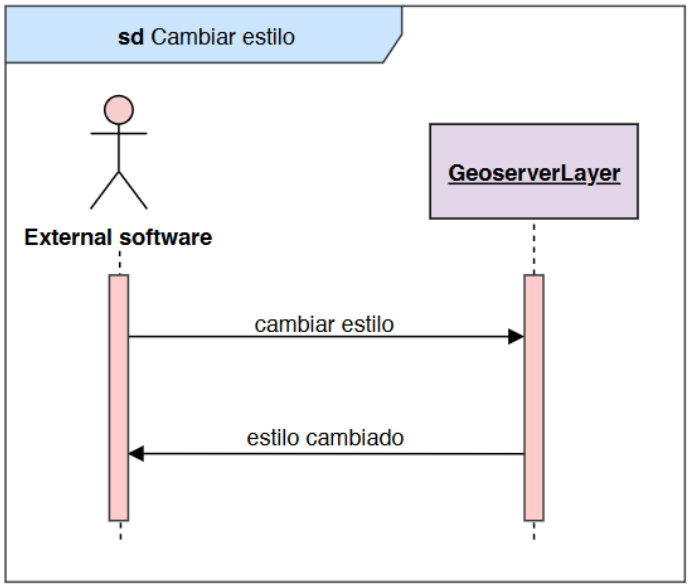


Figura 41. Diagrama de interacción para cambiar estilo.

Fuente: Elaboración propia

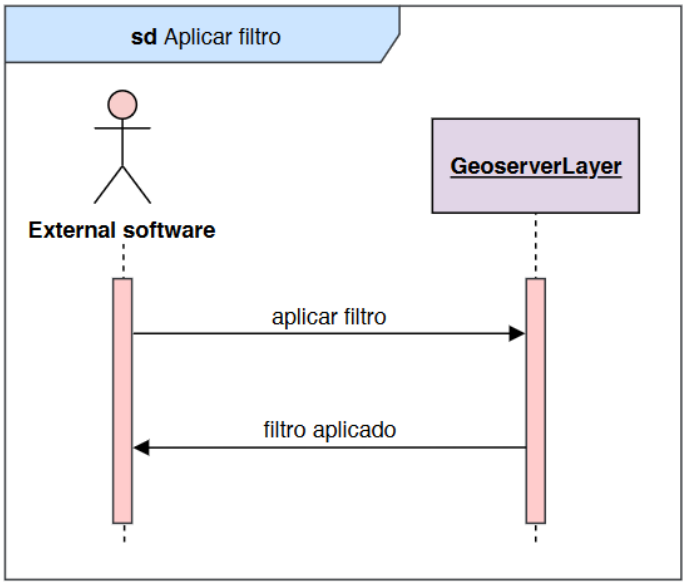


Figura 42. Diagrama de interacción para aplicar filtro.

Fuente: Elaboración propia

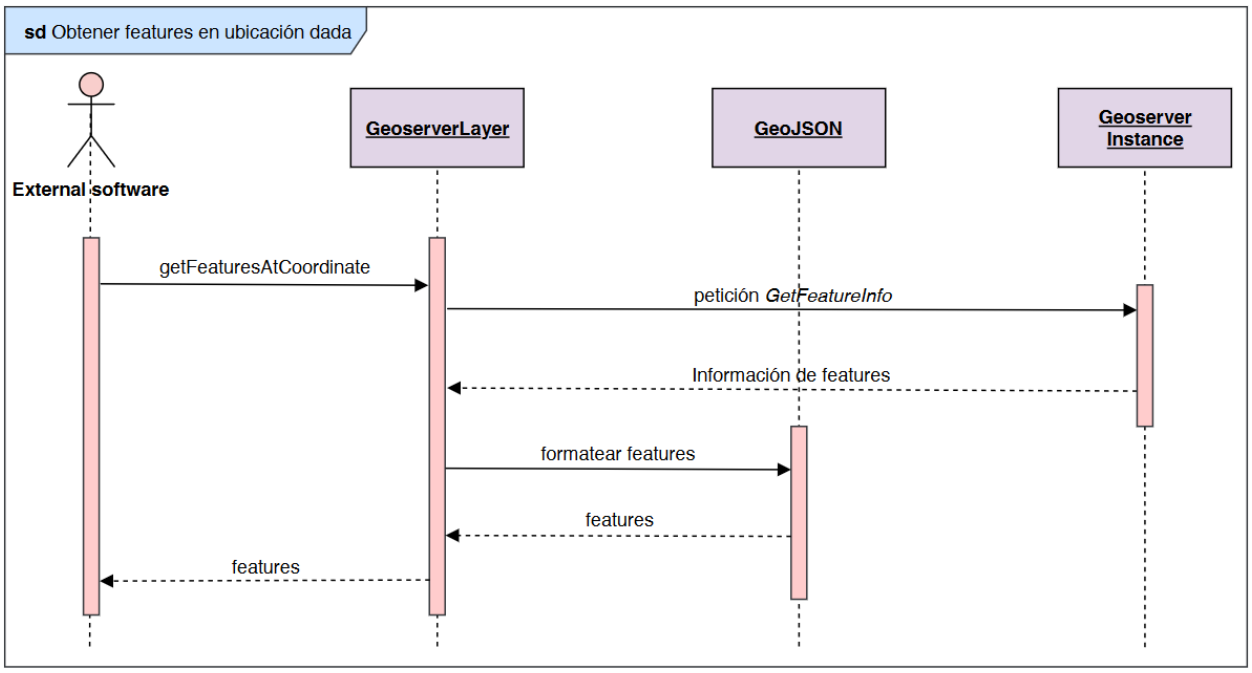


Figura 43. Diagrama de interacción para obtener features en ubicación dada.

Fuente: Elaboración propia

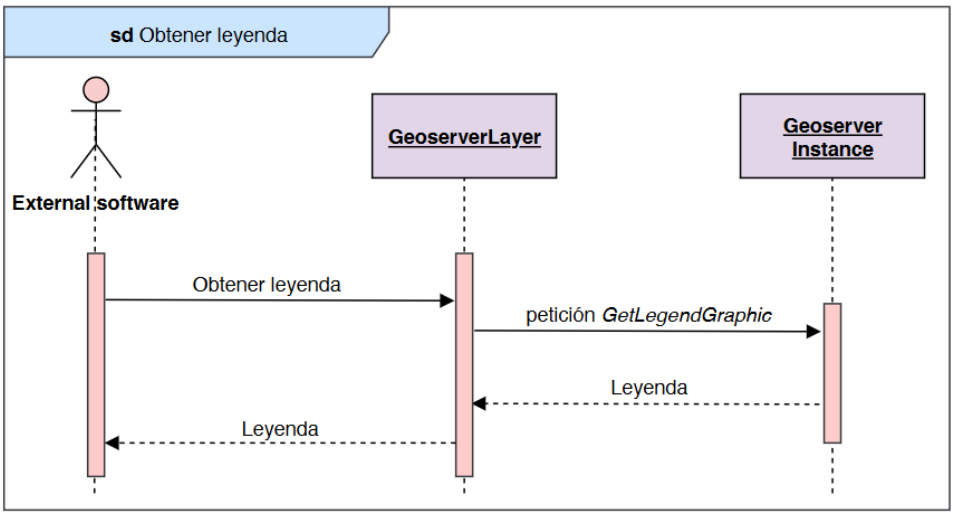


Figura 44. Diagrama de interacción para obtener leyenda.

Fuente: Elaboración propia

3.3.5.4. Módulo WFS

3.3.5.4.1. Diagrama de casos de uso

Se tomó en cuenta el hecho, de que con la operación GetFeature se puede obtener la información de los features, el número de features y además exportar los features en cualquiera de los formatos disponibles por Geoserver. La situación descrita, se puede observar en el diagrama representada como una generalización de casos de uso.

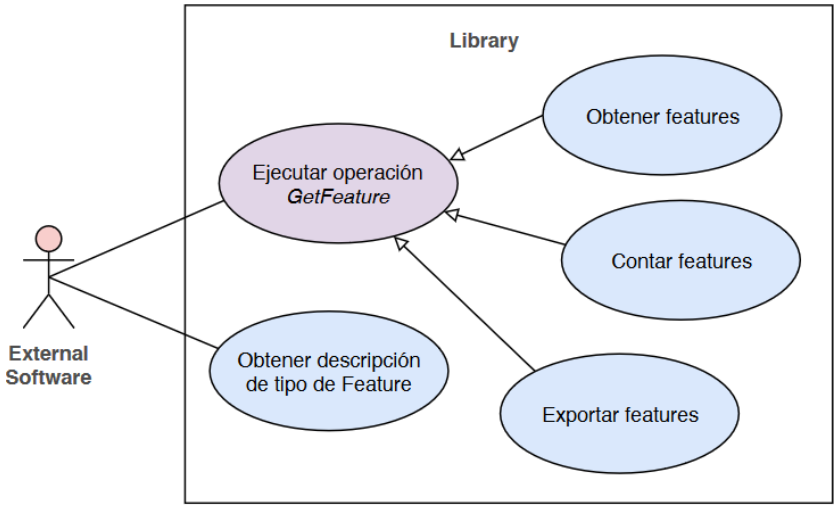


Figura 45. Diagrama de Clases de Uso del módulo WFS.

Fuente: Elaboración propia

A continuación, se detallan los casos de uso.

Detalle de caso de uso	
Nombre	Ejecutar operación *GetFeature*.
Actores	Software externo.
Función	Ejecutar una operación *GetFeature*.
Descripción	El software externo puede ejecutar una operación *GetFeature*.

Tabla 23. Detalle del caso de uso 'Ejecutar operación GetFeature'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Obtener features.
Actores	Software externo.
Función	Obtener features de una capa WMS.

Descripción	El software externo puede obtener features de las capas WMS en la forma de objetos de la clase *Feature*. Gracias a las propiedades que acepta la operación *GetFeature*, se puede restringir los features consultados.

Tabla 24. Detalle del caso de uso 'Obtener features'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Contar features.
Actores	Software externo.
Función	Contar features de una capa WMS.
Descripción	El software externo puede obtener el número de features de una capa. Gracias a las propiedades que acepta la operación *GetFeature*, se puede restringir los features consultados.

Tabla 25. Detalle del caso de uso 'Contar features'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Exportar features.
Actores	Software externo.
Función	Exportar features de una capa WMS.
Descripción	El software externo puede obtener una URL de exportación de los features de una capa. Los features podrán ser exportados en cualquier de los formatos soportados por GeoServer.

Tabla 26. Detalle del caso de uso 'Exportar features'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Obtener descripción del tipo de feature.
Actores	Software externo.
Función	Obtener descripción del tipo de feature de una capa WMS.
Descripción	El software externo puede obtener la descripción del tipo de feature de una capa WMS.

Tabla 27. Detalle del caso de uso 'Obtener descripción del tipo de feature'.

Fuente: Elaboración propia

3.3.5.4.2. Tipos de datos

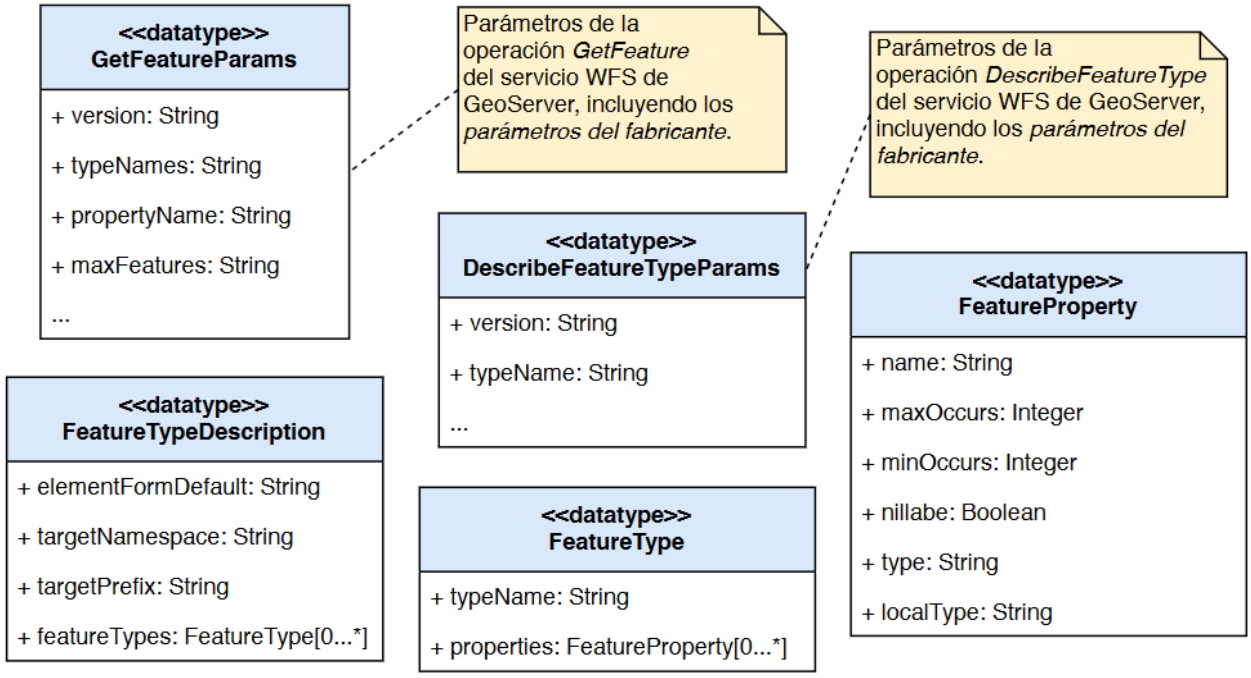


Figura 46. Tipos de datos del módulo WFS.

Fuente: Elaboración propia

3.3.5.4.3. Diagrama de clases

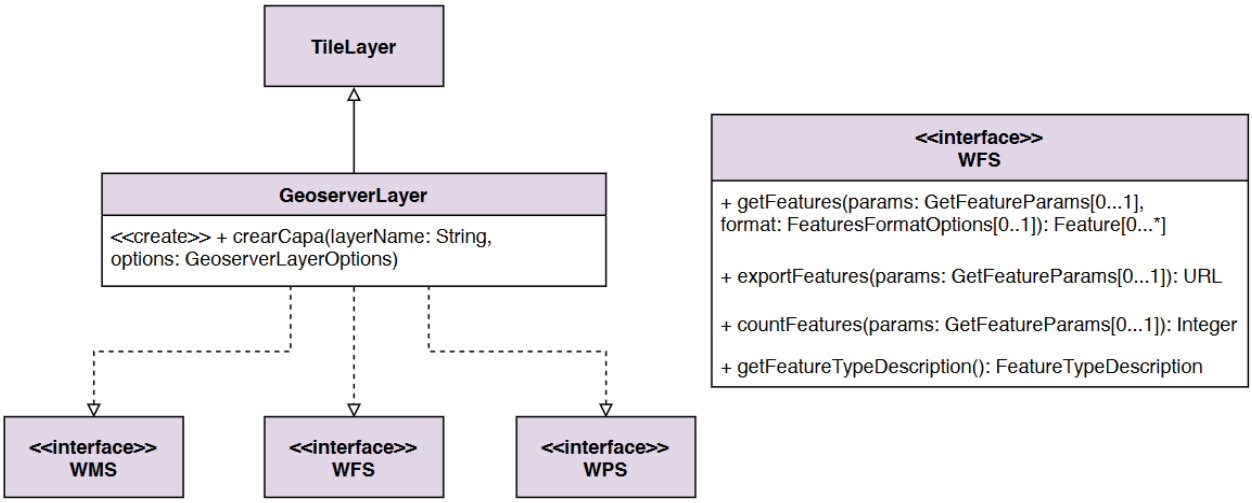


Figura 47. Diagrama de Clases del módulo WFS.

Fuente: Elaboración propia

3.3.5.4.4. Diagramas de interacción

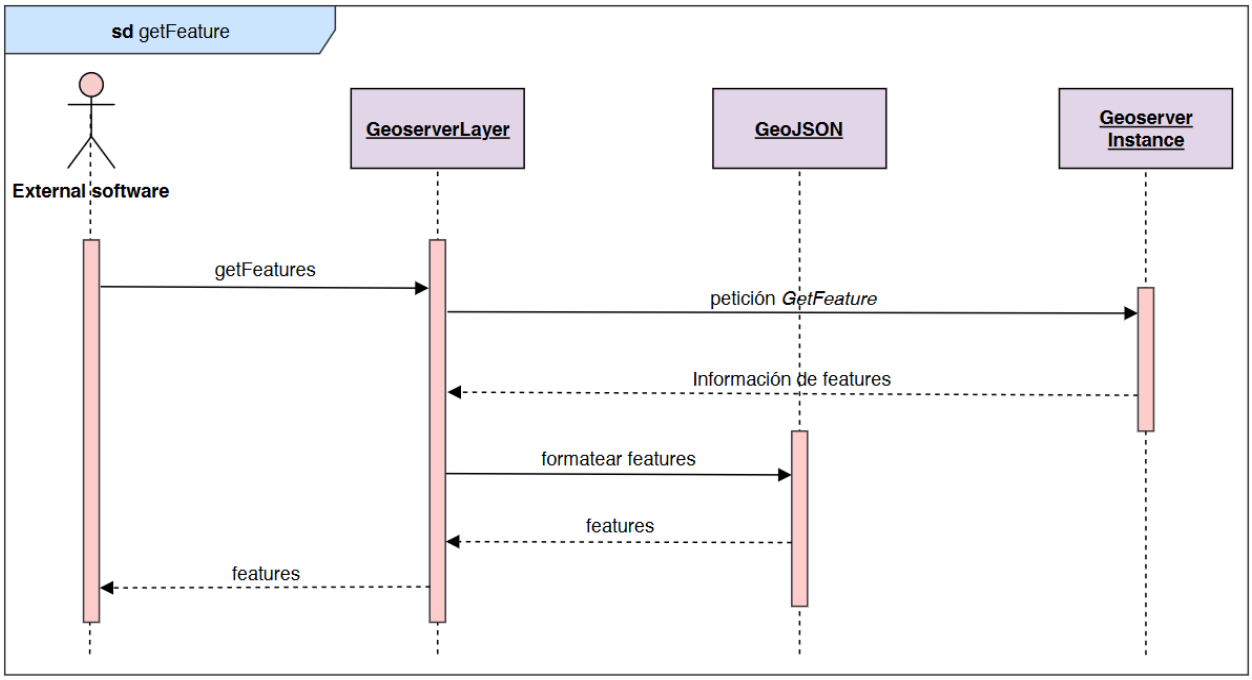


Figura 48. Diagrama de Interacción para la operación 'getFeatures'.

Fuente: Elaboración propia

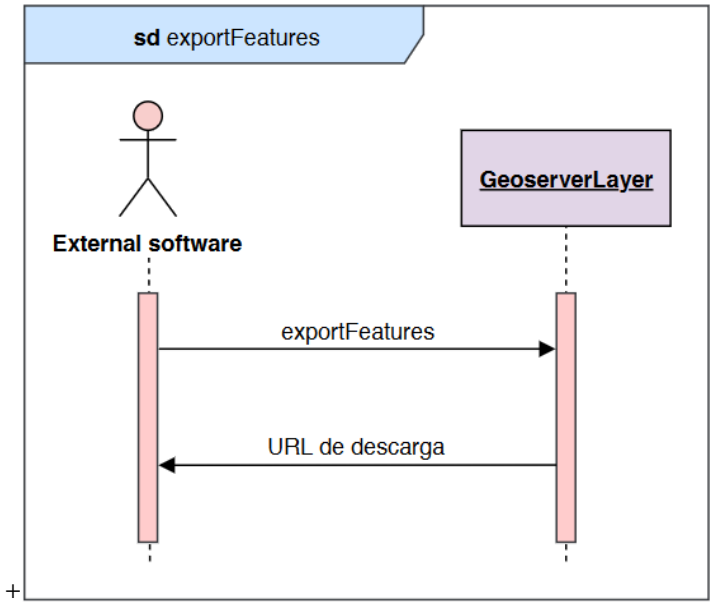


Figura 49. Diagrama de Interacción para la operación 'exportFeatures'.

Fuente: Elaboración propia

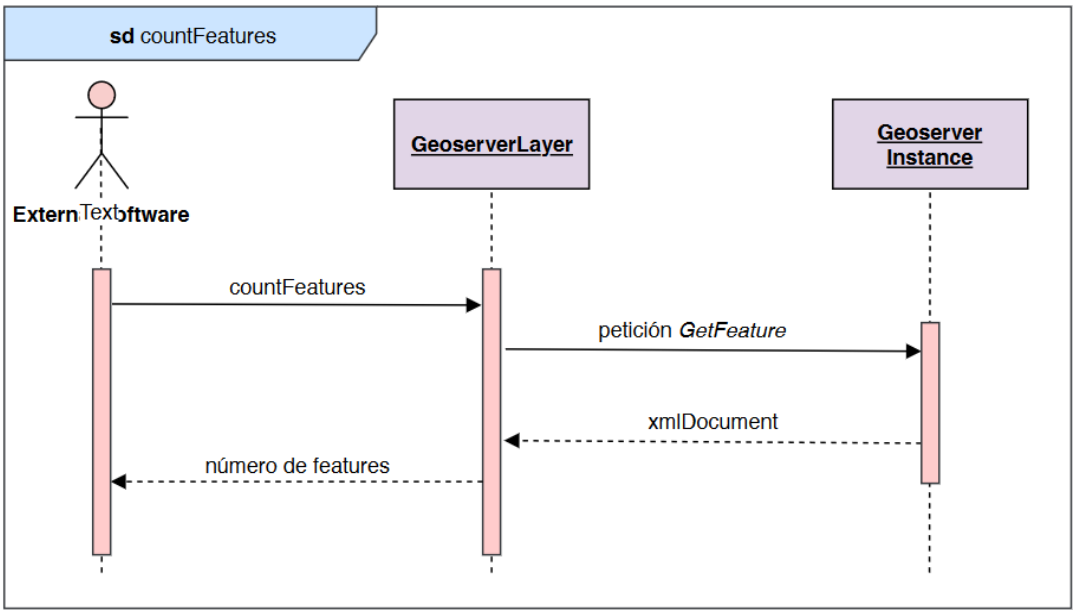


Figura 50. Diagrama de Interacción para la operación 'countFeatures'.

Fuente: Elaboración propia

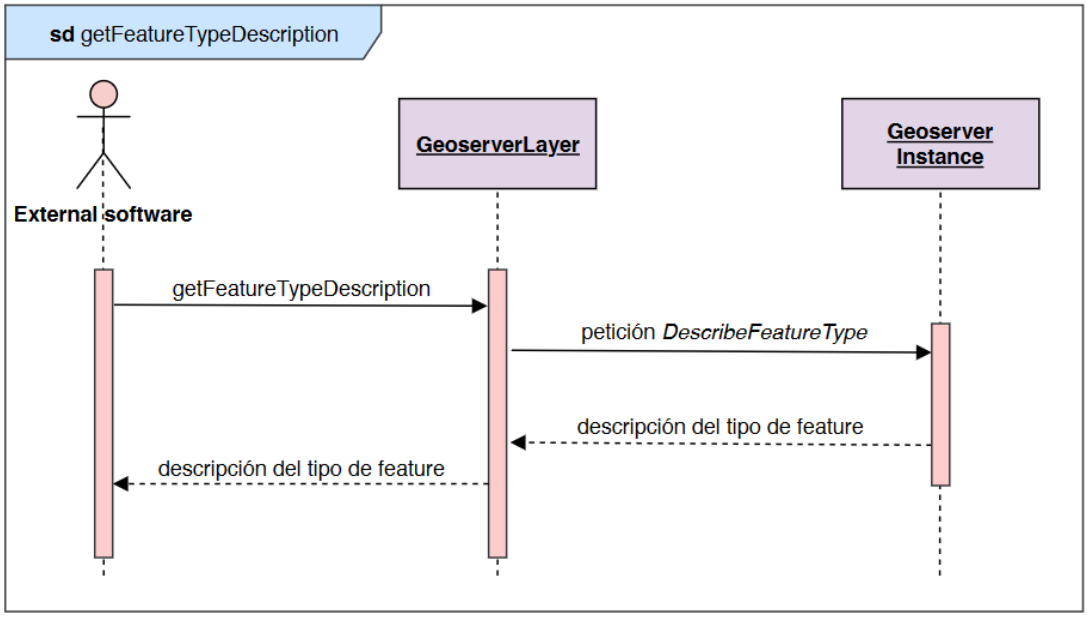


Figura 51. Diagrama de Interacción para la operación 'getFeatureTypeDescription'.

Fuente: Elaboración propia

3.3.5.5. Módulo WPS

3.3.5.5.1. Diagrama de casos de uso

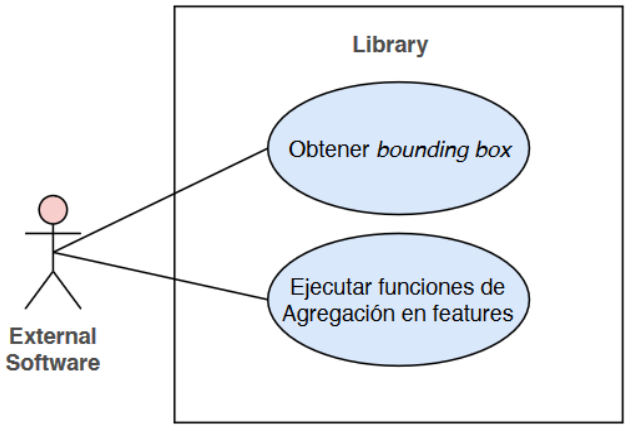


Figura 52. Diagrama de Casos de Uso del módulo WPS.

Fuente: Elaboración propia

A continuación, se detallan los casos de uso.

Detalle de caso de uso	
Nombre	Obtener *bounding box*.
Actores	Software externo.
Función	Obtener el bounding box de una capa WMS.
Descripción	El software externo puede obtener el bounding box de los features de una capa WMS. El resultado debe ser del tipo de dato *Extent*.

Tabla 28. Detalle del caso de uso 'Obtener bounding box'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Ejecutar funciones de agregación en features.
Actores	Software externo.
Función	Ejecutar funciones de agregación en features de capa WMS.
Descripción	El software externo puede ejecutar funciones de agregación sobres features de una capa WMS.

Tabla 29. Detalle del caso de uso 'Ejecutar funciones de agregación en features'.

Fuente: Elaboración propia

3.3.5.5.2. Tipos de datos

La definición de los datos, se hizo según los parámetros de los procesos *gs:Aggregate* y *gs:Bounds*. Cabe mencionar, que ambos procesos, precisan del parámetro *feature*, que sirve para definir los features para los cuáles los procesos serán ejecutados. Este parámetro, no se configura explícitamente, pues es tácito, que al ejecutar estas funciones desde el contexto de una instancia de la clase *GeoserverLayer*, que corresponde a una capa, los features a los que se aplicarán las operaciones son los features de la capa.

Por otro lado, se incluyó un parámetro *cql_filter*, que debe tomar el valor de una expresión ECQL, para que el usuario pueda precisar, del universo de features, un subconjunto al cuál se le aplicarán los procesos.

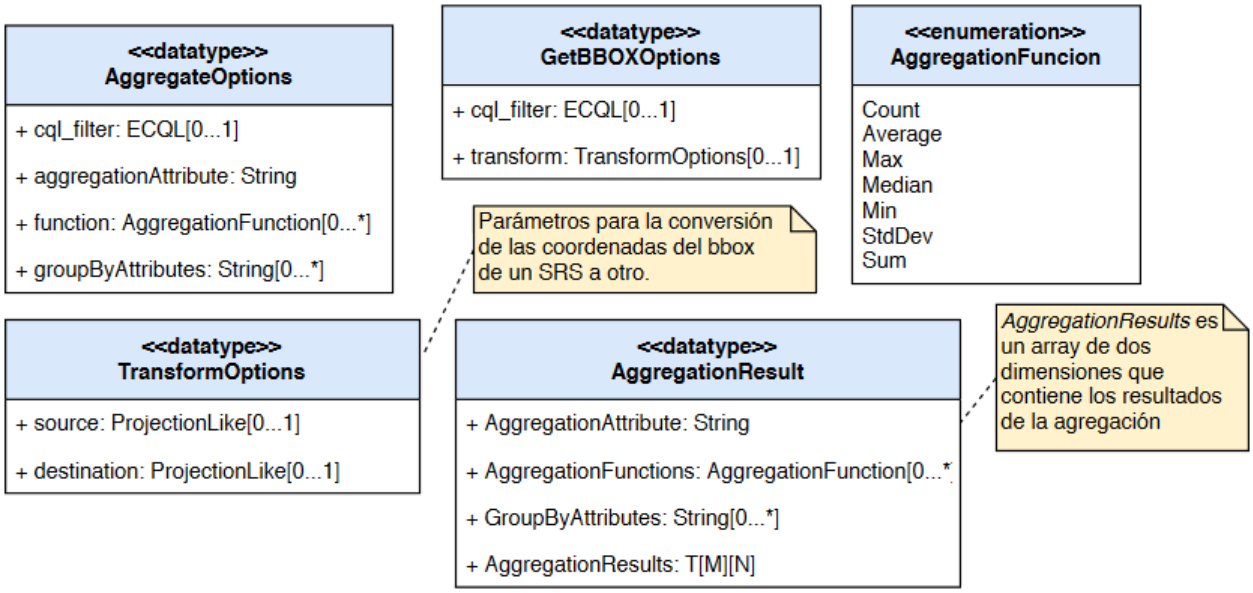


Figura 53. Tipos de datos del módulo WPS.

Fuente: Elaboración propia

Sobre las dimensiones de la propiedad 'AggregationResults', cabe precisar, que están en función de las propiedades 'groupByAttributes' y 'function' de 'AggregateOptions'. La relación existente va dada de la siguiente manera:

M es igual al número de clases resultantes de la agrupación, definida por el valor del parámetro 'groupByAttributes', y N es el igual al número funciones de agregación aplicadas sumado al número de atributos de agrupación.

3.3.5.5.3. Diagrama de clases

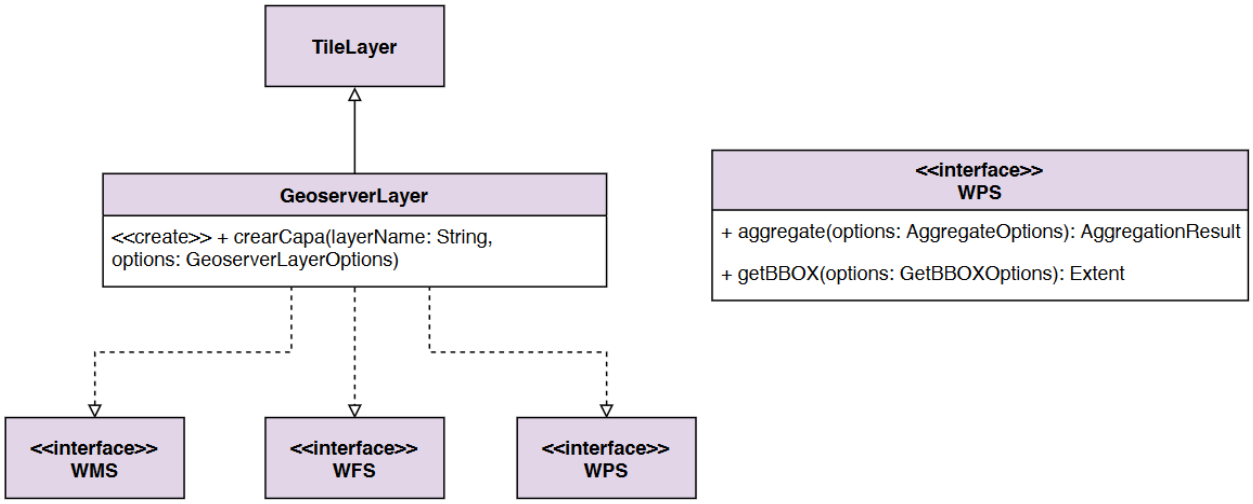


Figura 54. Diagrama de clases del módulo WPS.

Fuente: Elaboración propia

3.3.5.5.4. Diagramas de interacción

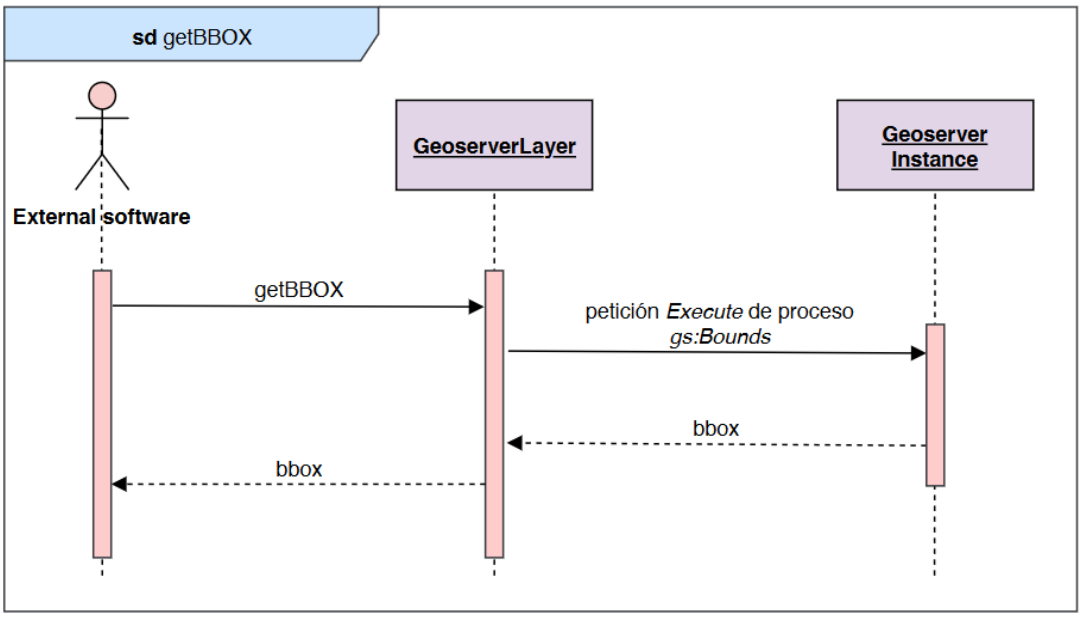


Figura 55. Diagrama de interacción para la operación 'getBBOX'.

Fuente: Elaboración propia

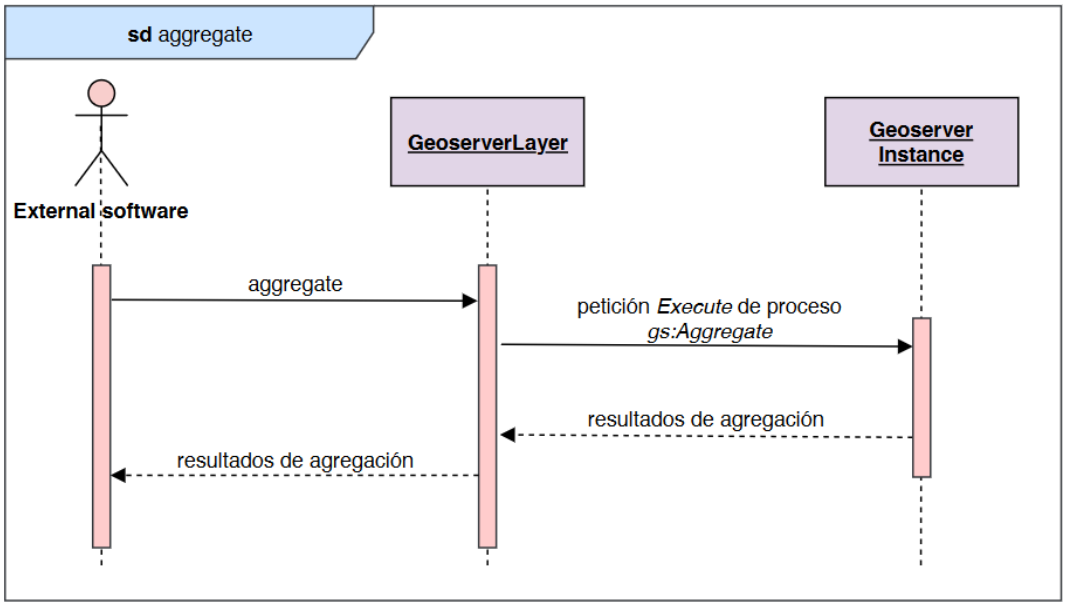


Figura 56. Diagrama de interacción para la operación 'aggregate'.

Fuente: Elaboración propia

3.3.5.6. Módulo OpenlayersLite

3.3.5.6.1. Diagrama de casos de uso

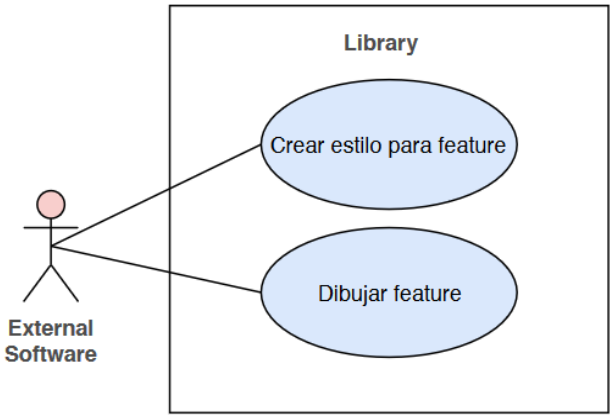


Figura 57. Diagrama de Casos de Uso del módulo OpenlayersLite.

Fuente: Elaboración propia

A continuación, se detallan los casos de uso.

Detalle de caso de uso	
Nombre	Crear estilo para feature.
Actores	Software externo.
Función	Crear estilo para un feature vectorial.
Descripción	El software externo puede crear un estilo aplicable a un feature vectorial.

Tabla 30. Detalle del caso de uso 'Crear estilo para feature'.

Fuente: Elaboración propia

Detalle de caso de uso	
Nombre	Dibujar feature.
Actores	Software externo.
Función	Dibujar un feature vectorial.
Descripción	El software externo puede dibujar un features vectorial, pudiendo ser, pero no limitándose, de los tipos de geometría: punto, línea y polígono.

Tabla 31. Detalle del caso de uso 'Dibujar feature'.

Fuente: Elaboración propia

3.3.5.6.2. Tipos de datos

Se definió el tipo de dato *StyleDefinition*, que comprende una serie de propiedades que en su conjunto definen un estilo para un feature, tales como: *fillColor* (color de relleno), *strokeColor* (color de borde), *strokeWidth* (ancho de borde), etc.

Relacionado a la función 'dibujar features', se definió el tipo de dato *DrawOptions*, que incluye parámetros para la creación de un objeto para dibujar features. Además, se definió un tipo de dato de nombre *SnapOptions*, que reúne propiedades que sirven para configurar el 'Snapping' (unión de features mientras se dibujan). Estas propiedades, son las mismas que se usan para crear un objeto de la clase ***Snap*** de Openlayers. Una definición detallada se puede encontrar en la documentación de Openlayers.

Se definieron también algunos tipos de enumeración, que especifican, los valores que pueden tomar algunas propiedades de los tipos de dato *StyleDefinition* y *SnapOptions*. El tipo *Shape* define los tipos de geometría que se pueden dibujar.

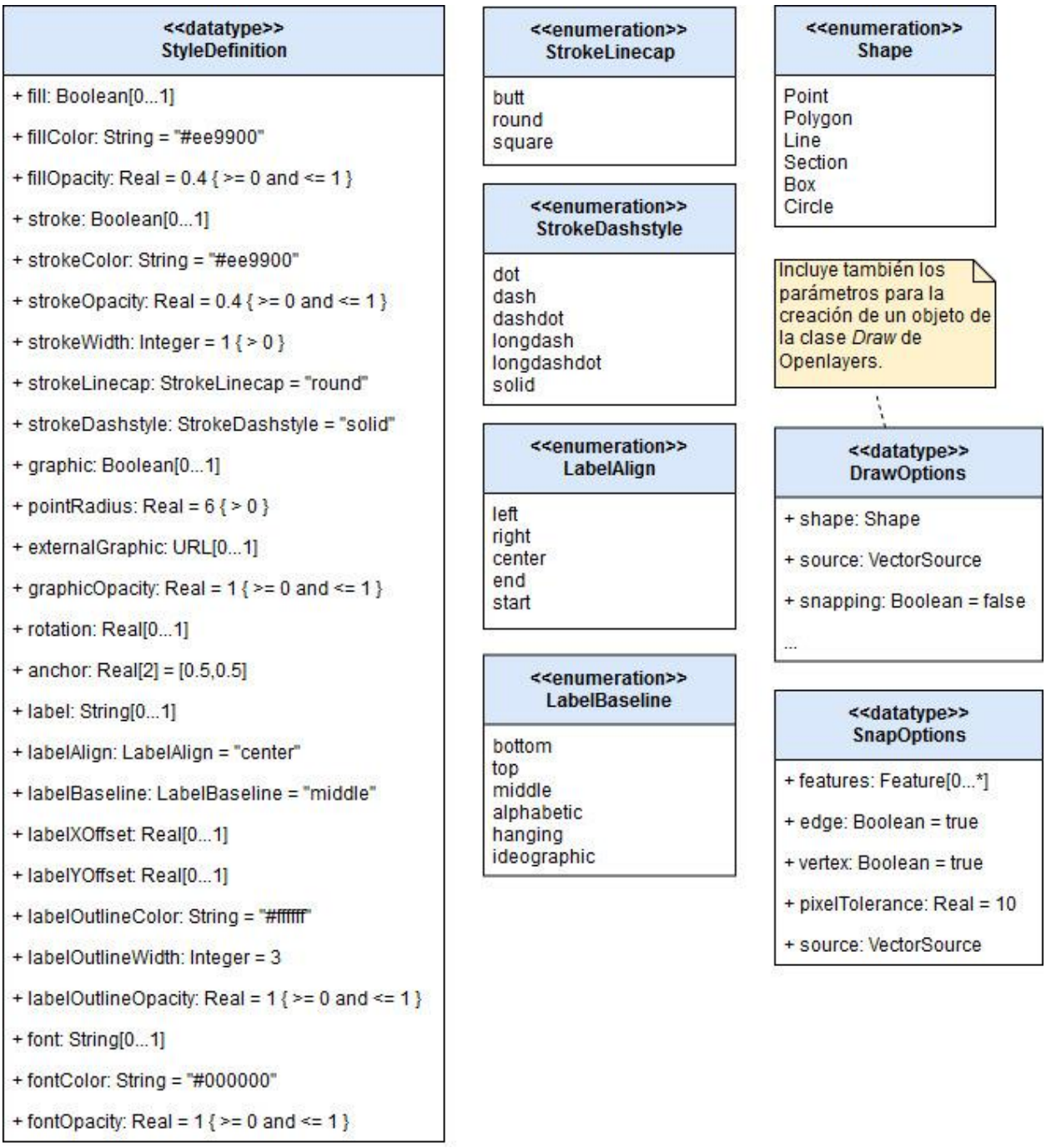


Figura 58. Tipos de datos del módulo OpenlayersLite.

Fuente: Elaboración propia

3.3.5.6.3. Diagrama de clases

Se concibieron las clases StyleLite y DrawLite para la creación de un estilo para un feature y el dibujo de un feature respectivamente. StyleLite, hereda de la clase Style de Openlayers, mientras que DrawLite de la clase Draw.

La clase DrawLite se modeló teniendo en cuenta que el dibujo de los features soportaría por defecto *Snapping*, por ello, se incluyó en el constructor un parámetro para la configuración del funcionamiento de dicha característica. La clase además aporta con los métodos *start* y *finish* para empezar y finalizar el dibujo respectivamente.

Dada la relación de herencia entre las clases Style y StyleLite, así como Draw y DrawLite, las primeras pueden ser usadas en lugar de las últimas (más no viceversa) pues cuenta con las mismas características. Esta situación viene a propósito, pues suscita la integración con Openlayers.

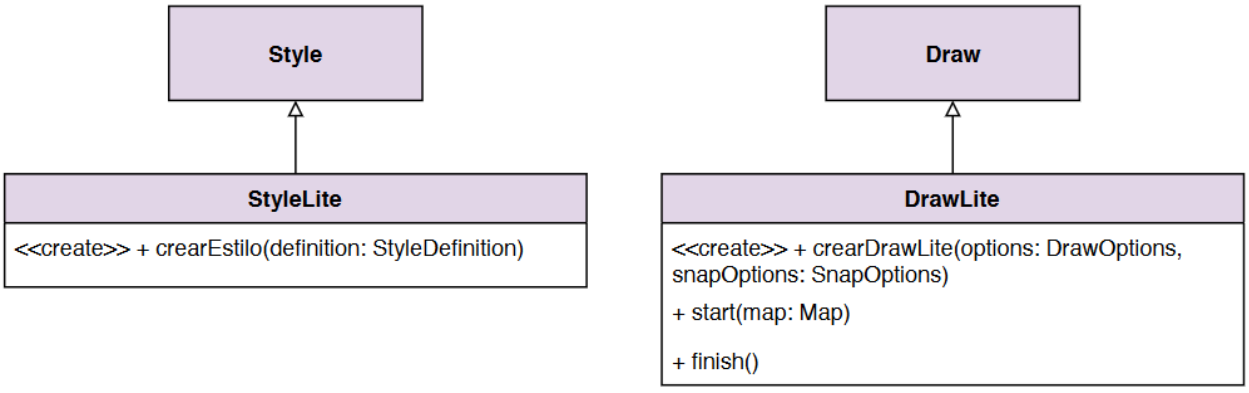


Figura 59. Diagrama de clases del módulo OpenlayersLite.

Fuente: Elaboración propia

3.3.5.6.4. Diagramas de interacción

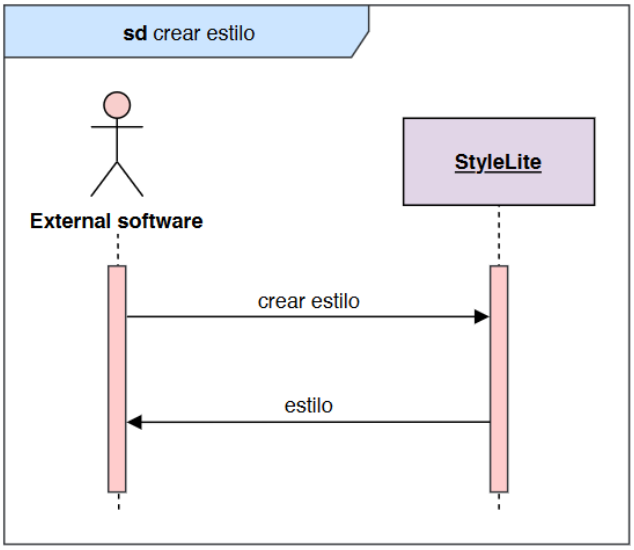


Figura 60. Diagrama de interacción para la operación 'crear un estilo'.

Fuente: Elaboración propia

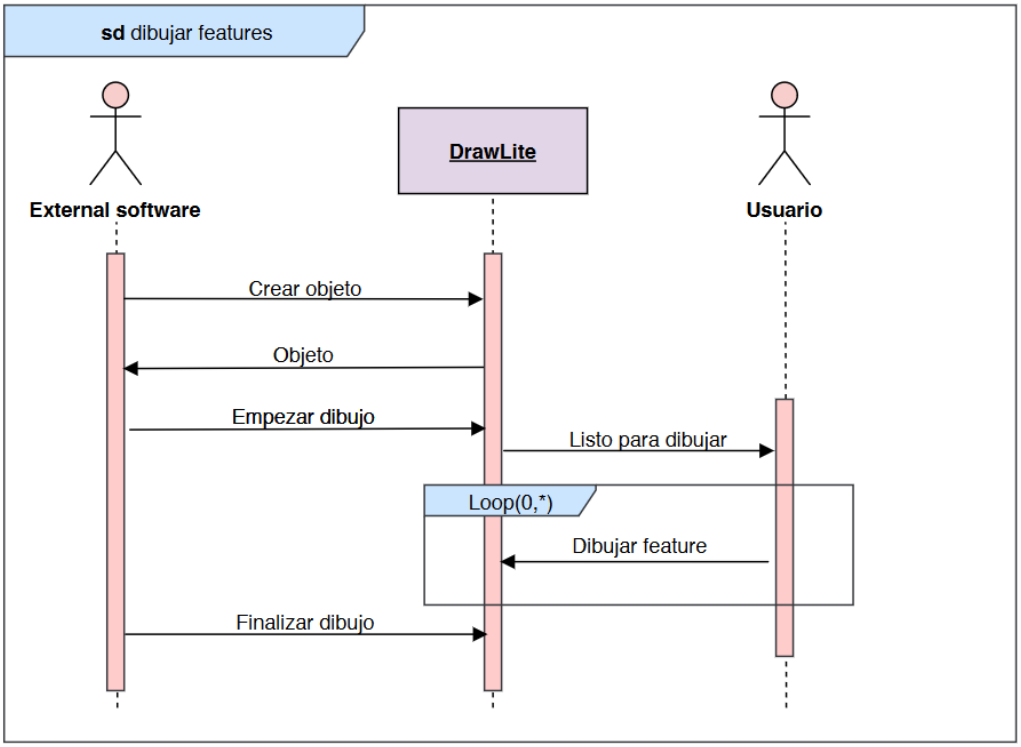


Figura 61. Diagrama de interacción para la operación 'dibujar features'.

Fuente: Elaboración propia

3.3.6. Desarrollo (codificación)

Se creó un repositorio en *github* para almacenar el código de la librería. El repositorio, fue creado bajo la licencia GNU 3.0 y es público. También, se creó un paquete en *npm*. Ambos, el repositorio y el paquete, fueron creados bajo el nombre de *easyolmaps*.

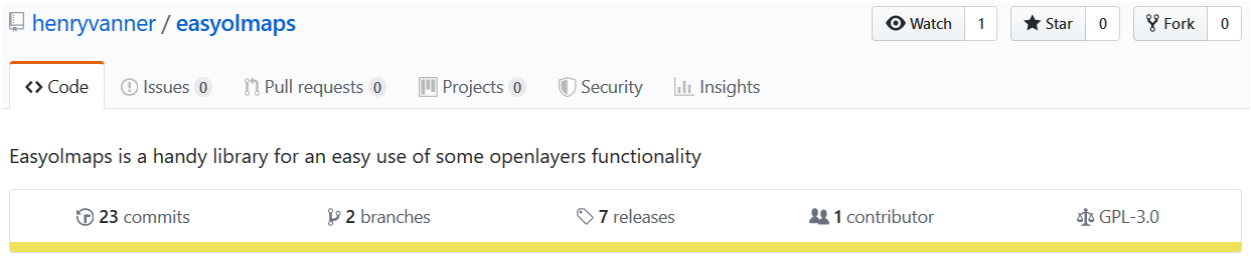


Figura 62. Repositorio en github.

Fuente: Elaboración propia

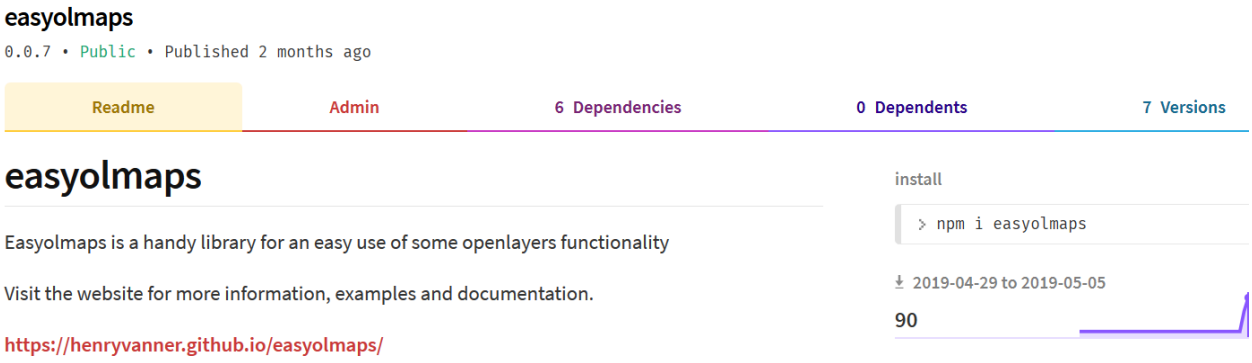


Figura 63. Paquete en npm.

Fuente: Elaboración propia

El desarrollo de la librería se realizó a la par con el desarrollo de una aplicación web sencilla, de modo que el avance y resultados se podían visualizar a través de ella. Las tecnologías que se utilizaron fueron las siguientes:

Ítem		Versión
JavaScript		ES6 y superiores
Openlayers		5.3.2
Geoserver	Geoserver	2.10.0
	WMS	1.1.1 y 1.3.0
	WFS	2.0.0, 1.1.0 y 1.0.0

	WPS	1.0.0
Node		10.15.3
Babel		7.4.4
React		16.8.3

Tabla 32. Tecnologías usadas para el desarrollo de la librería.

Fuente: Elaboración propia

En esta etapa, conjuntamente con el desarrollo de la librería, se elaboró la documentación en versión web. El sitio, fue publicado en internet con ayuda del servicio *GitHub Pages* de GitHub.

3.3.7. Evaluación de la calidad

Una vez desarrollada la librería, se aplicó el método de Baldeón Villanes (2015) para evaluar la calidad del producto. Cabe señalar que, aunque el método no lo requiere, en aras de dar mayor validez a los resultados, se realizaron tres evaluaciones, conducidas por cada uno de los evaluadores seleccionados. El proceso seguido en cada evaluación se relata a continuación.

En la fase 1 del método, se eligieron, a propósito de valorar el cumplimiento de los requerimientos funcionales y no funcionales de la librería, las características de la calidad que serían evaluadas. Estás se especificaron en el documento **'Requisitos de Evaluación de calidad'** (ANEXO 1) tal como lo requiere el método de evaluación aplicado, y también se describen a continuación.

Funcionalidad	
Completitud funcional	Grado en que el conjunto de funciones cubre todas las tareas y objetivos (requerimientos) especificados.
Compatibilidad	
Coexistencia	Grado en que un producto puede llevar a cabo sus funciones de manera eficiente mientras comparten un entorno y recursos con otros productos, sin impacto perjudicial en los otros productos.
Interoperabilidad	Grado en el cual dos o más sistemas, productos o componentes pueden intercambiar información y utilizar la información que se ha intercambiado.

Tabla 33. Características y sub-características de la calidad evaluadas. (Conforme al estándar ISO/IEC 25010).

Fuente: (Baldeón Villanes, 2015)

Elaboración propia

Usabilidad	
Simplificación del uso de Openlayers	Grado en que la librería simplifica el uso de las funcionalidades de Openlayers.

Tabla 34. Características particulares de la calidad evaluadas.

Fuente: Elaboración propia

En la fase 2, se especificaron las métricas que serían usadas para evaluar las características y sub-características de la calidad definidas en la fase 1, se definieron los criterios de decisión, así como la metodología que sería utilizada para la evaluación. Toda esta información se consignó en un documento titulado '**Especificaciones de Evaluación de la Calidad**' (ANEXO 2), tal como lo requiere el método de evaluación aplicado.

En la siguiente tabla, se describen las métricas usadas y los criterios de decisión.

ID	Nombre	Descripción	Característica evaluada	Función
CIF	Cobertura de implementación funcional	¿Qué tan completa es la implementación de acuerdo a las especificaciones de los requerimientos?	Completitud funcional	X = 1 – A/B, donde: A = Número de funciones que falta o están incorrectamente implementadas, que han sido detectadas en la evaluación. B = Número de funciones establecidas en la especificación de requerimientos.

Nota: Una función que falta o está incorrectamente implementada puede ser:

a) Función que no funciona según lo especificado en los manuales de usuario, especificación de requerimientos o especificaciones de diseño.

b) Función que no proporciona un resultado razonable y aceptable para lograr su objetivo.

COF	Coexistencia funcional	¿Qué parte de las funciones implementadas se ejecutan sin errores cuándo son usadas junto con Openlayers?	Coexistencia	X = A/B, dónde: A = Número de funciones que se ejecutaron correctamente y sin errores. B = Número de funciones que fueron implementadas con la librería y con Openlayers.
PII	Pertinencia para el intercambio de información	¿Qué parte de los tipos de datos de los parámetros de entrada y salida de las funciones de la librería tienen un formato que facilita el intercambio de información con Openlayers?	Interoperabilidad	X = A/B, dónde: A = Número de tipos de datos de parámetros que tienen un formato que facilita el intercambio de información con Openlayers. B = Número total de tipos de datos de los parámetros.

Nota: Se evalúan solo los tipos de datos que no son tipos de datos primitivos ni objetos JSON.

Un tipo de dato de un parámetro de entrada o salida de una función de la librería tienen un formato que facilita el intercambio de información si:

a) Es un objeto de Openlayers.
b) Es un objeto de la librería que está basado en un objeto Openlayers, tal que Openlayers lo puede utilizar sin producirse error alguno. Por ejemplo: los objetos de una clase que extiende de una clase de Openlayers.
c) Es un objeto de algunas de las APIs Web.

SNO	Simplificación del número de objetos	¿En cuánto disminuyó el número de objetos necesarios para usar una funcionalidad?	Simplificación del uso de Openlayers	X = B - A, dónde: A = Número de objetos necesarios para usar la funcionalidad con la librería. B = Número de objetos necesarios para usar la funcionalidad con Openlayers.
SNP	Simplificación del número de parámetros	¿En cuánto disminuyó el número de parámetros necesarios para usar una funcionalidad?	Simplificación del uso de Openlayers	X = B - A, dónde: A = Número de parámetros necesarios para usar la funcionalidad con la librería. B = Número de parámetros necesarios para usar la funcionalidad con Openlayers.

Tabla 35. Métricas usadas para evaluar la calidad.

Fuente: Elaboración propia

En la fase 3, se planificaron las actividades que se realizarían para efectuar la evaluación, y se representaron en un cronograma en el documento '**Diseño de la Evaluación de Calidad**' (ANEXO 3), tal como lo requiere el método de evaluación aplicado.

En la fase 4, se ejecutaron las actividades definidas en el cronograma. Se especificaron y diseñaron las pruebas que se realizaron para evaluar la calidad de la librería según los requisitos especificados en la fase 1 y las métricas detalladas en la fase 2. Las pruebas constaron básicamente en (1) el desarrollo de una aplicación web de mapas empleando la librería y (2) la implementación con la librería y Openlayers de un conjunto de tareas relacionadas a la creación de estilos para features vectoriales y dibujo de features. Las guías de observación (ANEXO 5, 6, 7 y 8) se usaron para obtener los datos y producir los resultados que se incluirían en el informe final de evaluación.

3.4.TÉCNICAS E INSTRUMENTOS

Para la evaluación de la calidad, se hicieron uso de guías de observación para recoger datos sobre las pruebas ejecutadas y características de la librería, que sirvieron para la aplicación de las métricas y para producir los resultados sobre la calidad de la librería.

Indicador	Técnica	Instrumento
Cobertura de implementación funcional	Observación	Guía de observación Nº 1. (ANEXO N° 5)
Coexistencia funcional	Observación	Guía de observación Nº 2. (ANEXO N° 6)
Pertinencia para el intercambio de la información	Observación	Guía de observación Nº 3. (ANEXO N° 7)
Simplificación del uso	Observación	Guías de observación Nº 4. (ANEXO N° 8)

Tabla 36. Técnicas e Instrumentos.

Fuente: Elaboración propia

3.5.ASPECTOS ÉTICOS

En el presente trabajo de investigación, que tiene como objetivo principal el desarrollo de una librería (producto de software) basada en Openlayers para aplicaciones web de mapas, la misma que se sometió a una evaluación de calidad, no existió falsificación alguna de datos ni tergiversación de los resultados de ningún tipo.

La información usada durante el desarrollo de la librería y para su posterior evaluación, que no pertenecen al investigador, fueron utilizadas previo consentimiento de los propietarios, únicamente para el propósito de servir para el desarrollo y evaluación de la librería, y según los parámetros de uso aprobado establecidos por los propietarios.

Las herramientas que se usaron para el desarrollo y para su posterior evaluación, tales como, programas para la edición de código, sistemas de control de versiones, otras librerías, etc., son en su mayoría de uso libre u OpenSource y aquellas que no lo son, tal es el caso (y solo el único) del programa estadístico SPSS, fue usado dentro del período de prueba, tal que no se incurrió en el uso ilegal ni piratería de ningún software.

CAPÍTULO IV. RESULTADOS Y DISCUSIONES

4.1.RESULTADOS

4.1.1. Evaluación de la calidad

Antes de presentar los resultados, es pertinente señalar que, desde antes de ejecutar las evaluaciones, ya se esperaban los mismos resultados, pues a pesar de que fueron conducidas por distintas personas, las condiciones y especificaciones de las pruebas fueron las mismas.

4.1.1.1. Completitud funcional

Para evaluar la completitud funcional, recordamos que este se puede medir con ayuda del indicador de **cobertura de implementación funcional** (CIF). La prueba consistió en el desarrollo de una aplicación web con la librería para constatar que esta cumpliera con los requisitos funcionales especificados. Los resultados obtenidos se muestran en la tabla a continuación:

Variable	Valor
Número de funciones que faltan o están incorrectamente implementadas, que han sido detectadas en la evaluación.	0
Número de funciones establecidas en la especificación de requerimientos.	11
CIF = 100%	

Tabla 37. Resultados completitud funcional.

Fuente: Elaboración propia

4.1.1.2. Coexistencia

Con respecto a la coexistencia, que es el grado en que la librería y Openlayers pueden trabajar juntos sin impacto perjudicial en su funcionamiento, recordamos que se puede estimar con el indicador de **coexistencia funcional** (COF). Para ello, se evaluó de la aplicación web desarrollada, el correcto funcionamiento de aquellas funciones para cuya implementación se precisó del uso de la librería junto con Openlayers. Los resultados obtenidos se muestran a continuación:

Variable	Valor
Número de funciones que se ejecutaron correctamente y sin errores.	5
Número de funciones que fueron implementadas con la librería y con Openlayers.	5
COF = 100%	

Tabla 38. Resultados coexistencia.

Fuente: Elaboración propia

4.1.1.3. Interoperabilidad

Para evaluar la interoperabilidad, se puede usar el indicador de **pertinencia para el intercambio de la información** (PII), que señala el grado en que los tipos de datos de los parámetros de entrada y salida de las funciones de la librería tienen un formato que facilita el intercambio de información con Openlayers. Para ello, fue preciso la evaluación de cada uno de los tipos de datos de la librería según las pautas especificadas en la tabla N° 35 correspondientes a PII. Los resultados obtenidos se muestran a continuación:

Variable	Valor
Número de tipos de datos de parámetros que tienen un formato que facilita el intercambio de información con Openlayers.	10
Número total de tipos de datos de los parámetros.	10
PII = 100%	

Tabla 39. Resultados Interoperabilidad.

Fuente: Elaboración propia

4.1.1.4. Simplificación del uso de Openlayers

Las funcionalidades que se quieren someter a evaluación son dos: (1) la creación de estilos para features vectoriales y (2) el dibujo de features vectoriales. Lo que se quiere estimar, es el grado en que la librería simplifica el uso de dichas funcionalidades, es decir, que tan fácil es crear un estilo para un feature vectorial o dibujar un feature usando la librería, que si se usara Openlayers.

Para tal efecto, se puede recurrir a la diferencia de objetos y parámetros que se necesitan para usar dichas funciones, entre la librería y Openlayers Así, para la evaluación se plantearon una serie de tareas relacionadas a ambas funciones que debían ser realizadas con la librería y Openlayers por separado. Es decir, por cada tarea se debían hacer dos implementaciones, una usando la librería y otra usando Openlayers, de modo que se puedan obtener dos indicadores: (1) la diferencia entre el número objetos que se necesitan usando Openlayers (NOO) y el número de objetos que se necesitan usando la librería (NOL), y (2) la diferencia entre el número de parámetros que se necesitan usando Openlayers (NPO) y el número de parámetros que se necesitan usando la librería (NPL). Los resultados obtenidos se muestran a continuación:

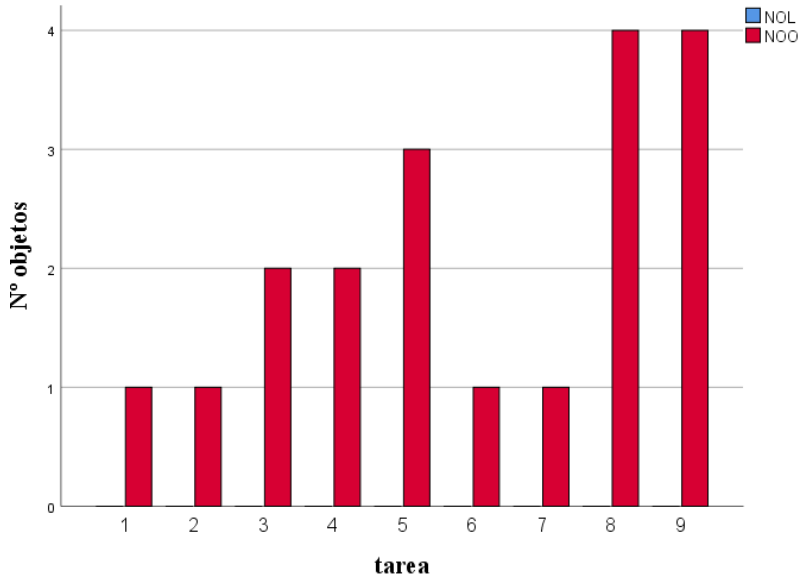


Figura 64. Número de objetos que necesitan la librería y Openlayers para la creación de estilos para features vectoriales.

Fuente: Elaboración propia

En el gráfico de barras presentado en la figura 64, referente a NOL y NOO para la realización de tareas relacionadas a la creación de estilos para features vectoriales, se observa que, cuando se usa la librería, el número de objetos que se necesitan para realizar las tareas es 0.

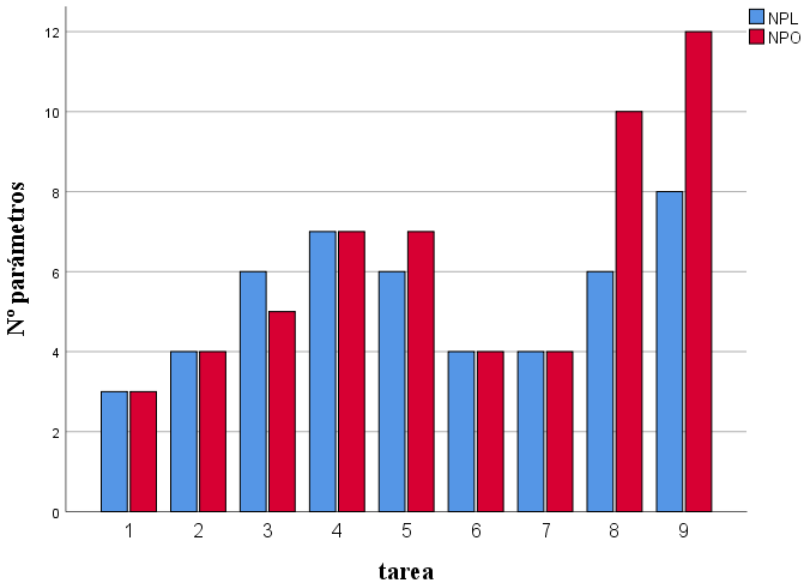


Figura 65. Número de parámetros que necesitan la librería y Openlayers para la creación de estilos para features vectoriales.

Fuente: Elaboración propia

En el gráfico de barras presentado en la figura 65, referente a NPL y NPO para la realización de tareas relacionadas a la creación de estilos para features vectoriales, se observa que, para la mayoría de tareas, NPL es menor o igual que NPO, excepto para la tarea 3, donde su valor supera al valor de NPO. El siguiente gráfico revela más información sobre la diferencia entre NPO y NPL.

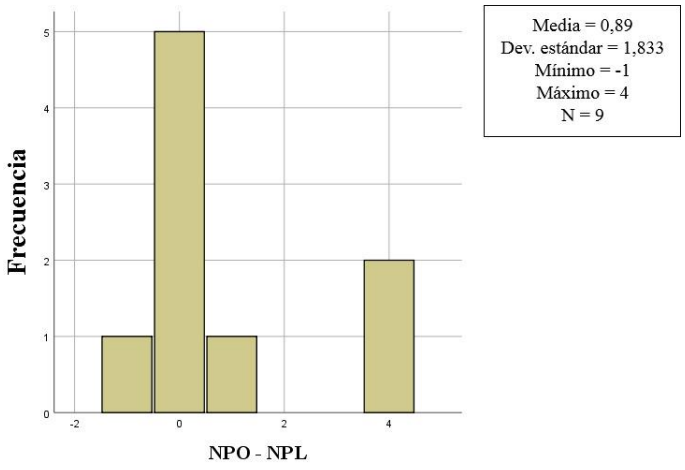


Figura 66. Histograma de la diferencia entre NPO y NPL para la creación de estilos para features vectoriales.

Fuente: Elaboración propia

En el histograma que se presenta en la figura 66, referente a la diferencia entre NPO y NPL, se observa que esta diferencia varía entre -1 y 4 parámetros, y que la diferencia más frecuente fue de 0 parámetros, seguida de la diferencia de 4 parámetros.

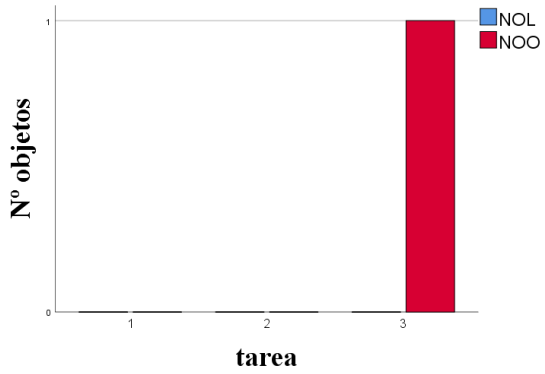


Figura 67. Número de objetos que necesitan la librería y Openlayers para el dibujo de features vectoriales.

Fuente: Elaboración propia

En el gráfico de barras presentado en la figura 67, referente a NOL y NOO para la realización de tareas relacionadas al dibujo de features vectoriales, se observa que, cuando se usa la librería, el número de objetos que se necesitan para realizar las tareas es 0.

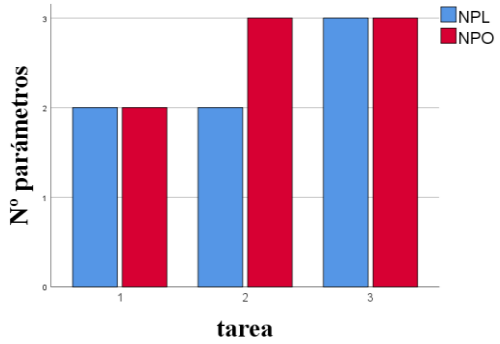


Figura 68. Número de parámetros que necesitan la librería y Openlayers para el dibujo de features vectoriales.

Fuente: Elaboración propia

En el gráfico de barras presentado en la figura 68, referente a NPL y NPO para la realización de tareas relacionadas al dibujo de features vectoriales, se observa que, para las tareas 1 y 3 la diferencia fue de 0, y para la tarea 2 fue de 1 parámetro.

4.1.2. Aplicación web de mapas

A continuación, se muestran capturas de pantalla de la aplicación web que se desarrolló para la evaluación de la librería.

En las imágenes mostradas de la aplicación se pueden apreciar dos capas ráster que fueron facilitadas por la empresa EPS GRAU S.A., que cuenta con un servidor GeoServer versión 2.10.0.

Alias	**Nombre (GeoServer)**	**Estilos disponibles**
Tuberías de agua	sig_tuberias_agua_2	▪ tuberia_agua_2 ▪ tuberia_agua_por_tipo_2
Habilitaciones urbanas	sig_urbanismo_2	▪ sig_urbanismo_2
URI del servicio WMS: http://gisteco.epsgrau.pe:8080/geoserver/web/wms		

Tabla 40. Capas ráster de la empresa EPS GRAU S.A. usadas para probar la librería

Fuente: Elaboración propia

La aplicación web tiene una UI que se divide en dos partes: (1) UI para la prueba de las funciones de los módulos de la librería, y (2) Mapa interactivo.

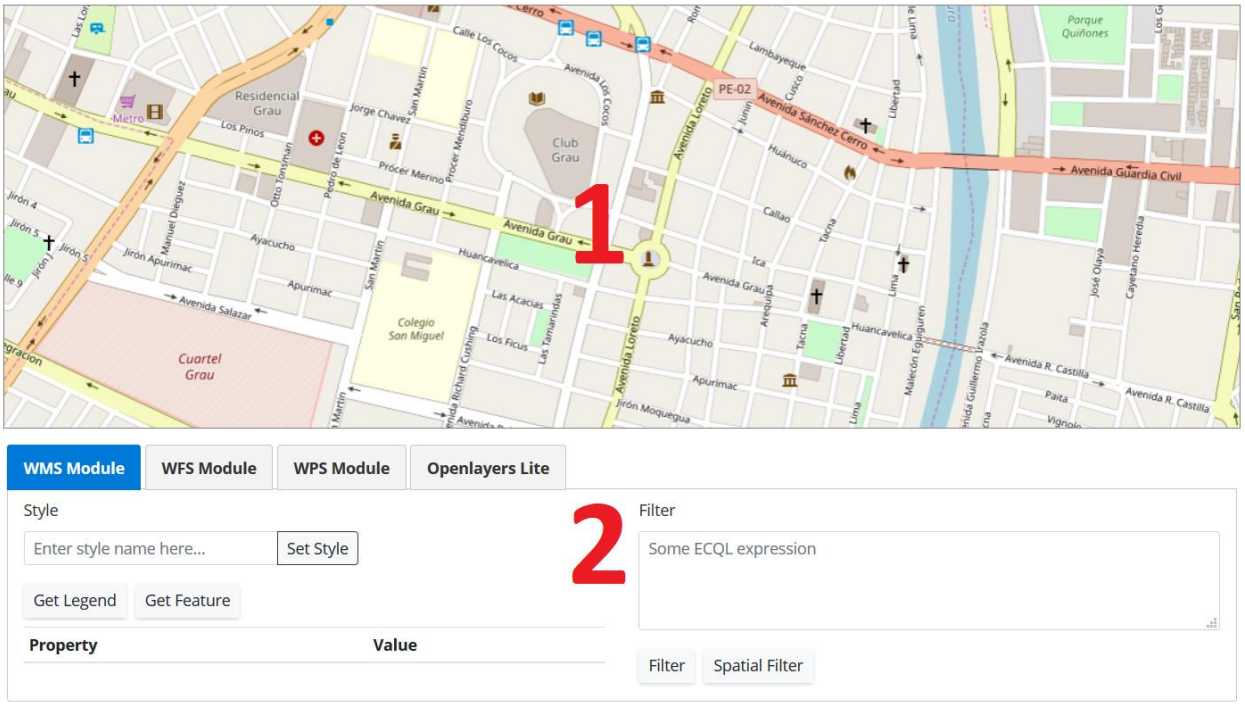


Figura 69. Aplicación web desarrollada para la evaluación de la librería.

Fuente: Elaboración propia

4.1.2.1. Módulo WMS

La UI para la prueba de las funcionalidades del módulo WMS, está constituida de los siguientes elementos:

1. Caja de texto para ingresar una expresión ECQL.
2. Botón para filtrar el contenido de la capa.
3. Caja de texto para ingresar el nombre de un estilo.
4. Botón para cambiar el estilo de la capa.
5. Botón para ver la leyenda de la capa.
6. Botón para para seleccionar un elemento de la capa y ver su información.
7. Espacio para mostrar la información del elemento de la capa seleccionado.
8. Botón para dibujar un polígono y filtrar los elementos de la capa que se encuentren dentro de él.

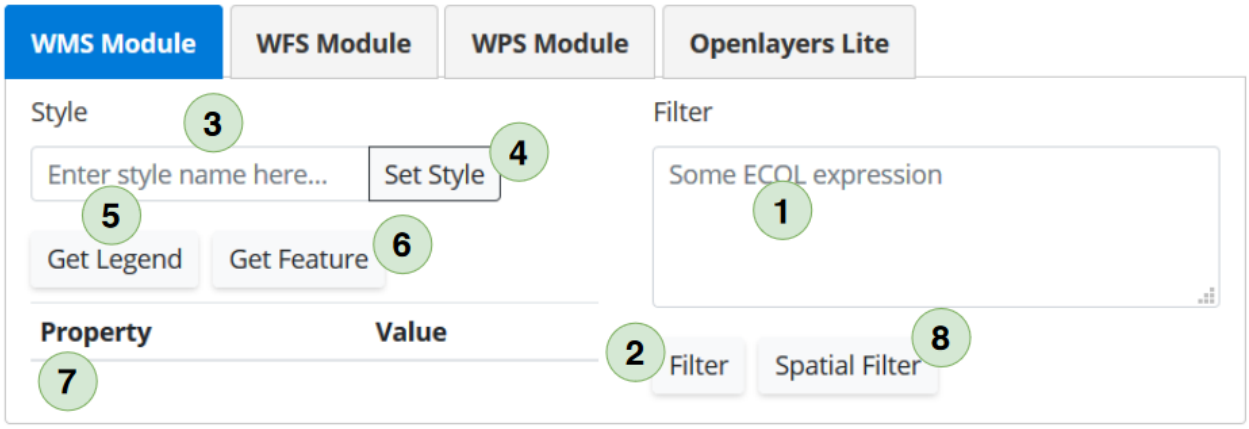


Figura 70. UI para la prueba de las funcionalidades del módulo WMS.

Fuente: Elaboración propia

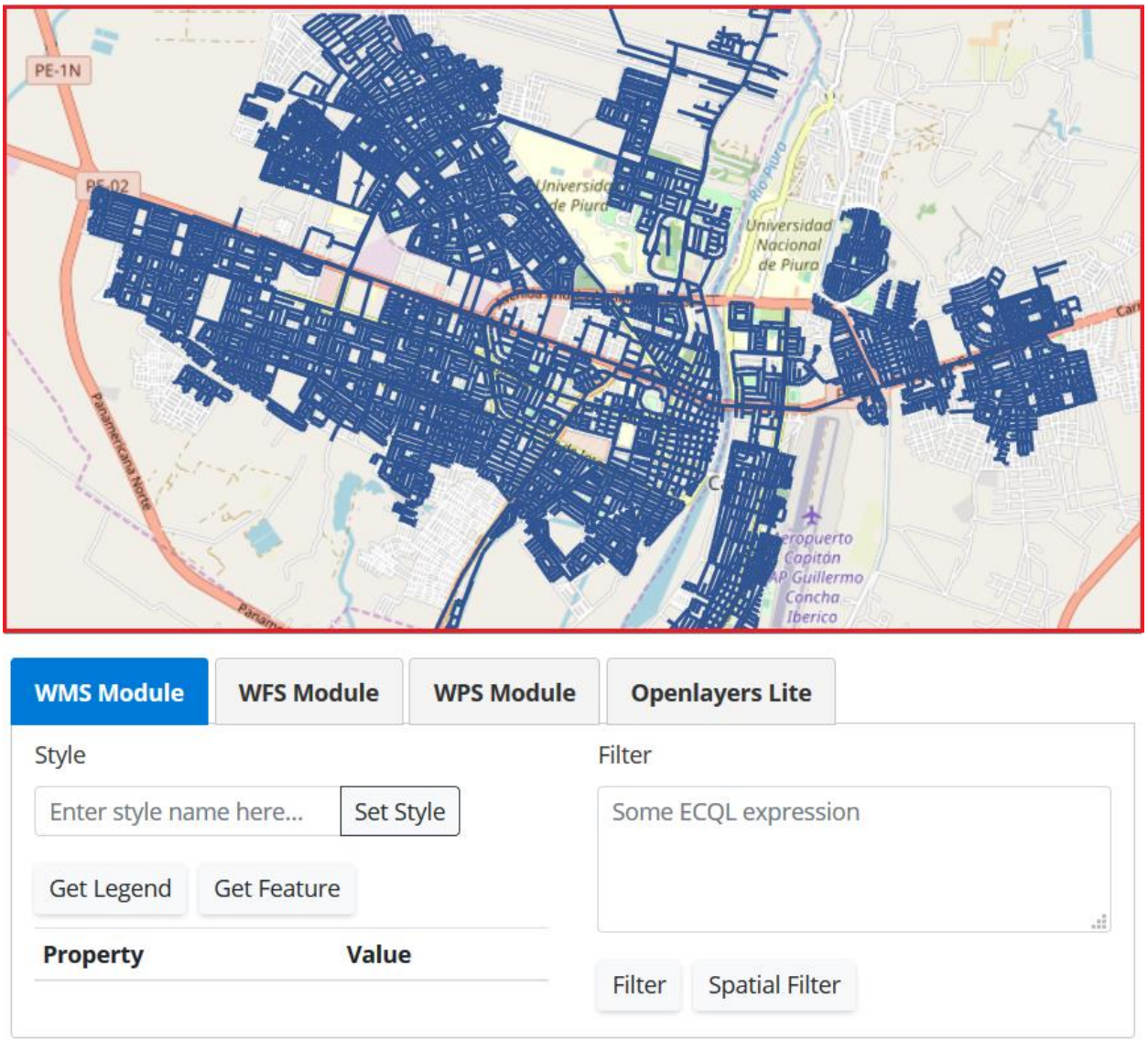


Figura 71. Visualización de la capa Tuberías de agua.

Fuente: Elaboración propia

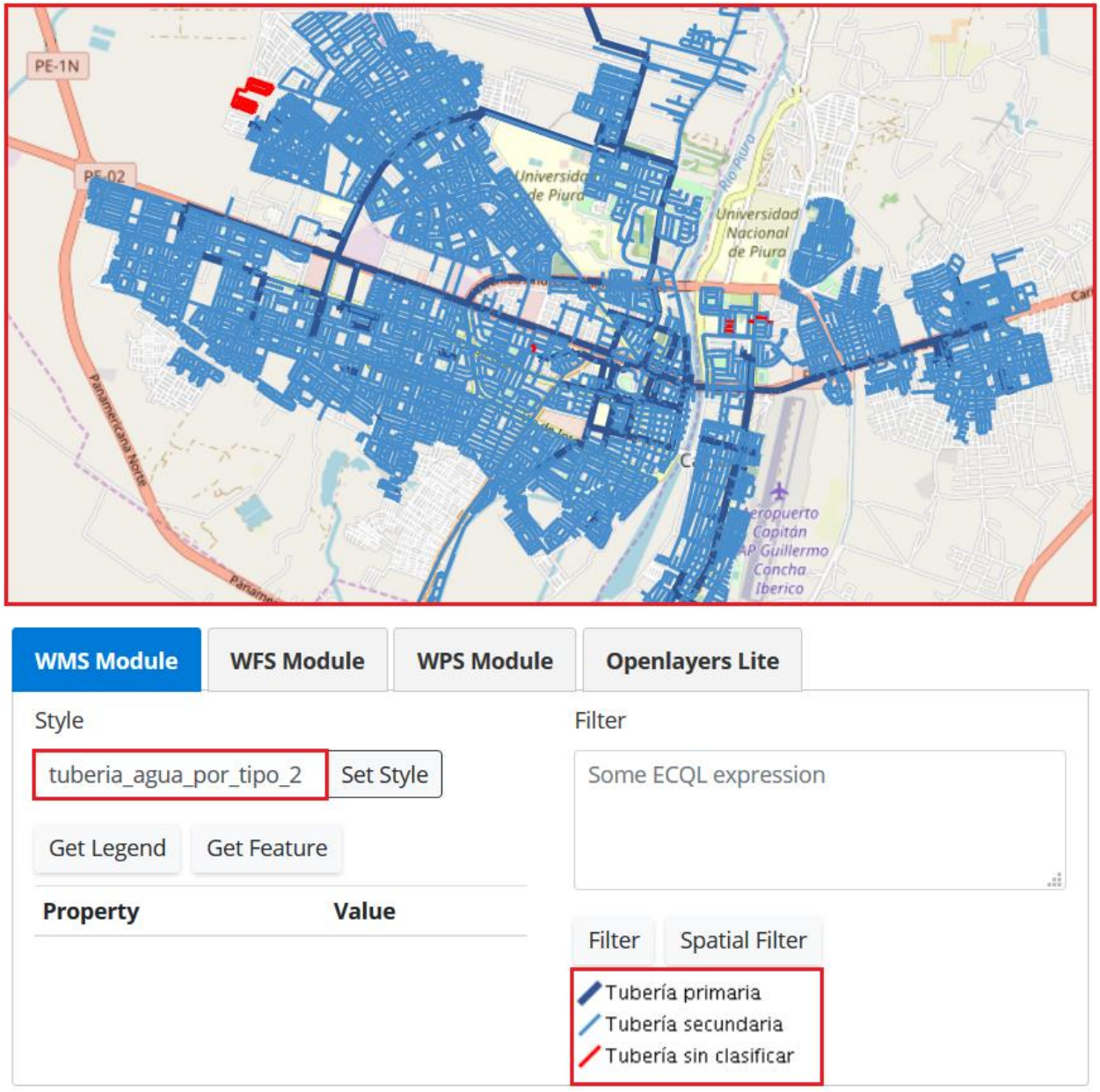


Figura 72. Capa Tuberías de agua con estilo 'tuberia_agua_por_tipo_2' y la leyenda correspondiente.

Fuente: Elaboración propia

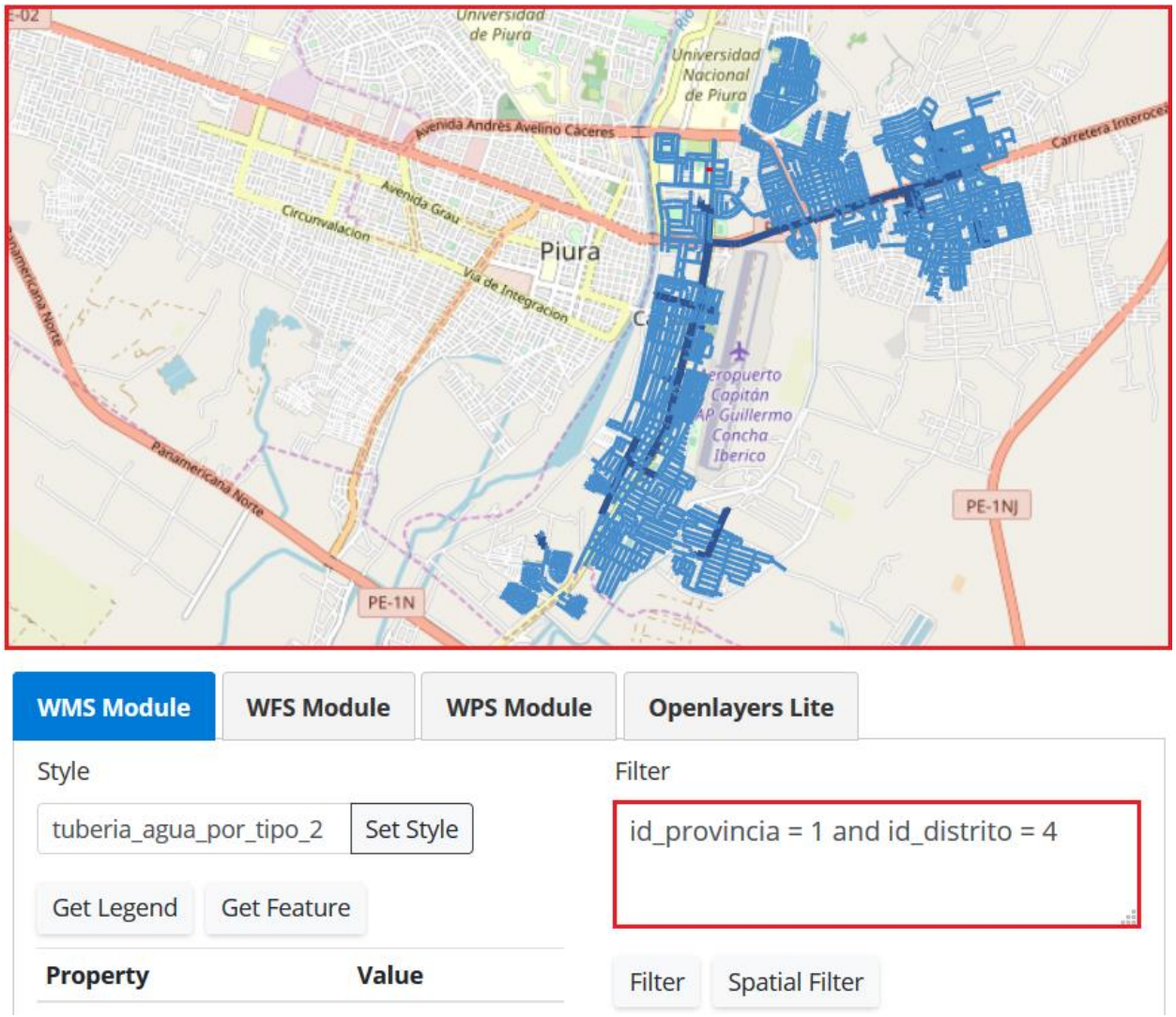
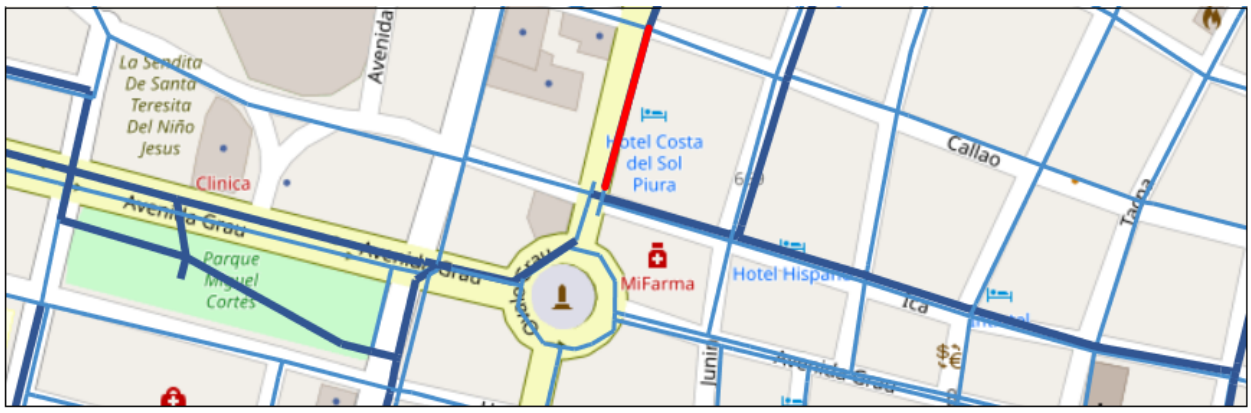


Figura 73. Capa Tuberías de agua filtrada para mostrar solo las tuberías del distrito de castilla.

Fuente: Elaboración propia



WMS Module | WFS Module | WPS Module | Openlayers Lite

Style

tuberia_agua_por_tipo_2 | Set Style

Get Legend | Get Feature

Filter

Some ECQL expression

Filter | Spatial Filter

Property	Value
sin_etiqueta	
length	115.06
length_gis	108.264549752722
node1	643-C1
node2	637-V1
dn_plg	10
dn_mm	250
material	PVC

Figura 74. Información de una tubería seleccionada.

Fuente: Elaboración propia

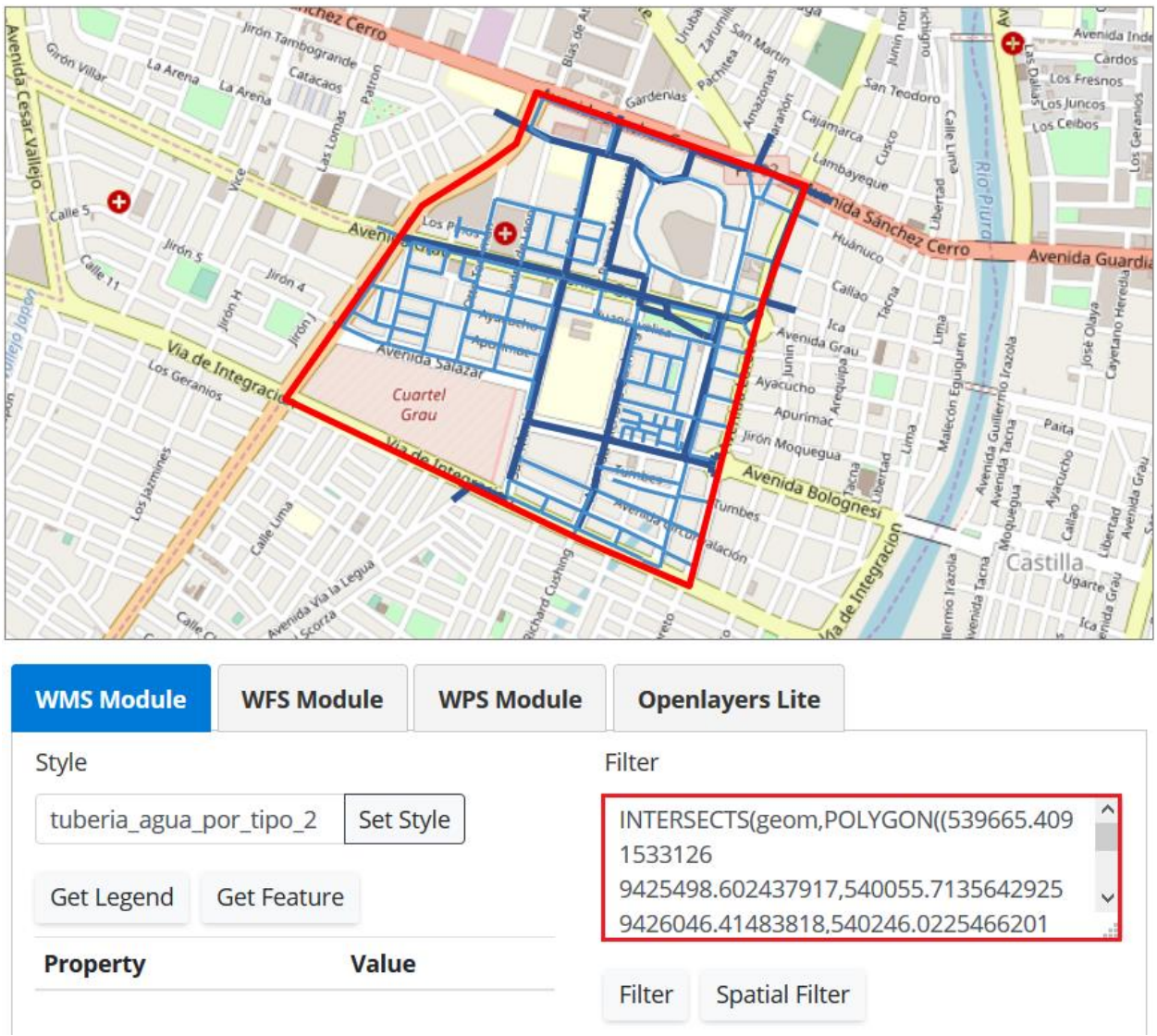


Figura 75. Filtrado de tuberías que se encuentra dentro del área de un polígono dibujado.

Fuente: Elaboración propia

4.1.2.2. Módulo WFS

La UI para la prueba de las funcionalidades del módulo WFS, está constituida de los siguientes elementos:

1. Botón para contar los elementos.
2. Botón para obtener los features de la capa y mostrarlos en el mapa.
3. Opción para obtener los features de la capa y mostrarlos en el mapa.
4. Espacio para mostrar la descripción del tipo de feature de la capa.
5. Botón para exportar los features de la capa.

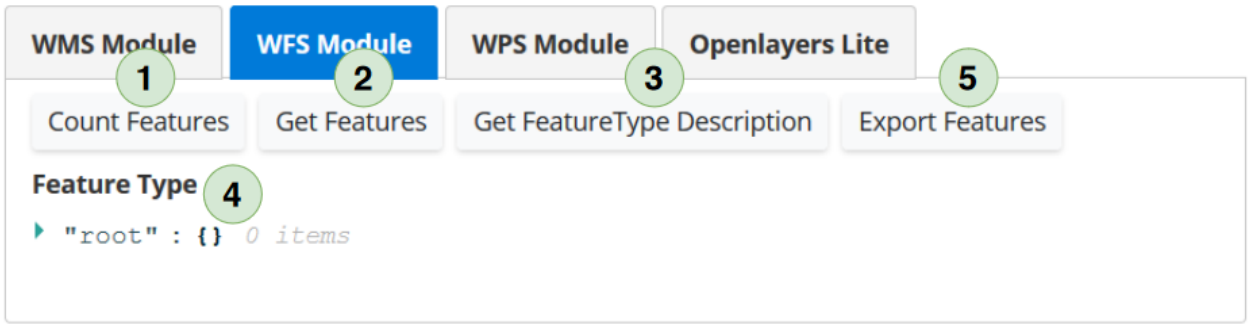


Figura 76. UI para la prueba de funcionalidades del módulo WFS.

Fuente: Elaboración propia

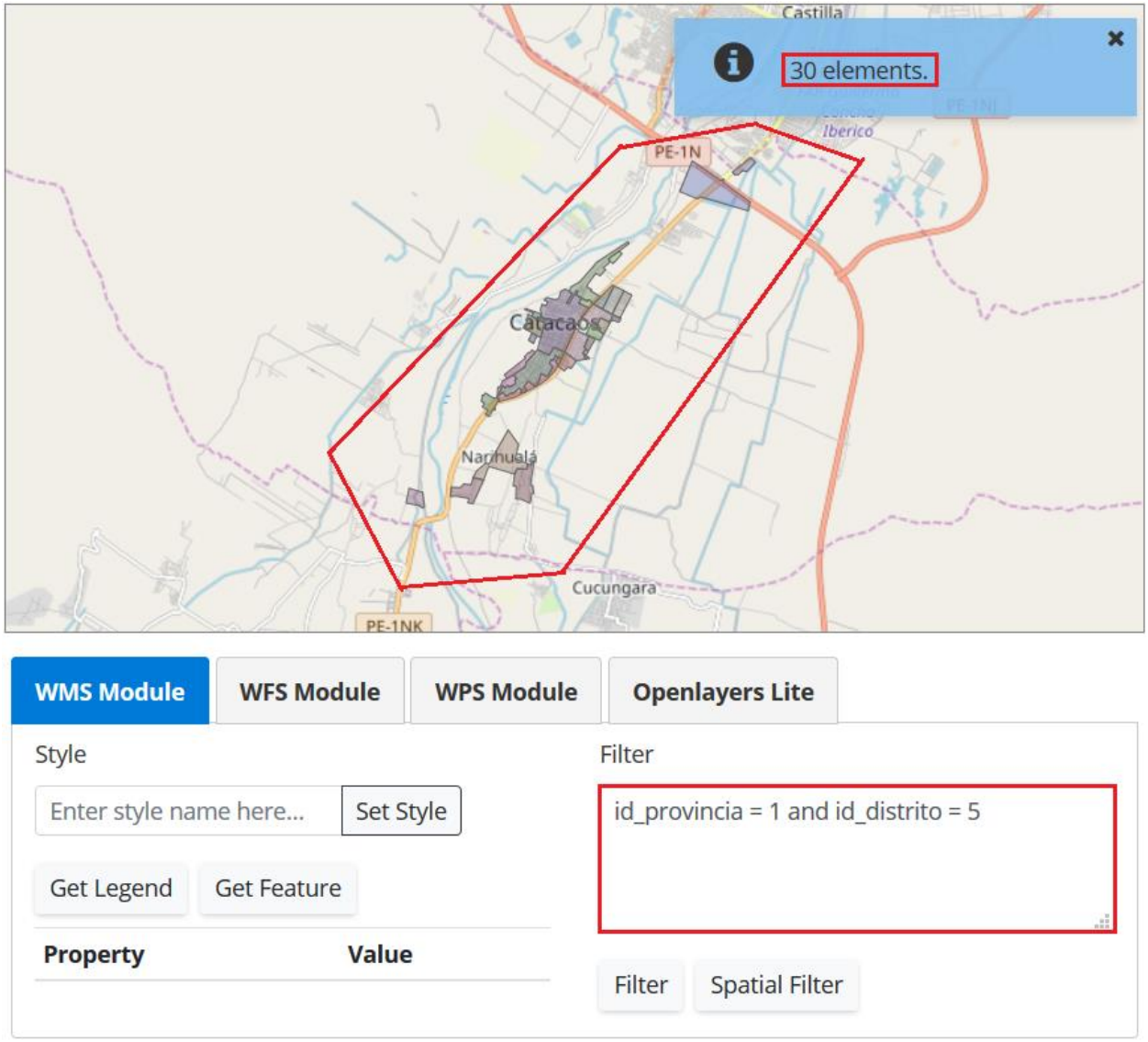


Figura 77. Número de habilitaciones urbanas del distrito de Catacaos.

Fuente: Elaboración propia

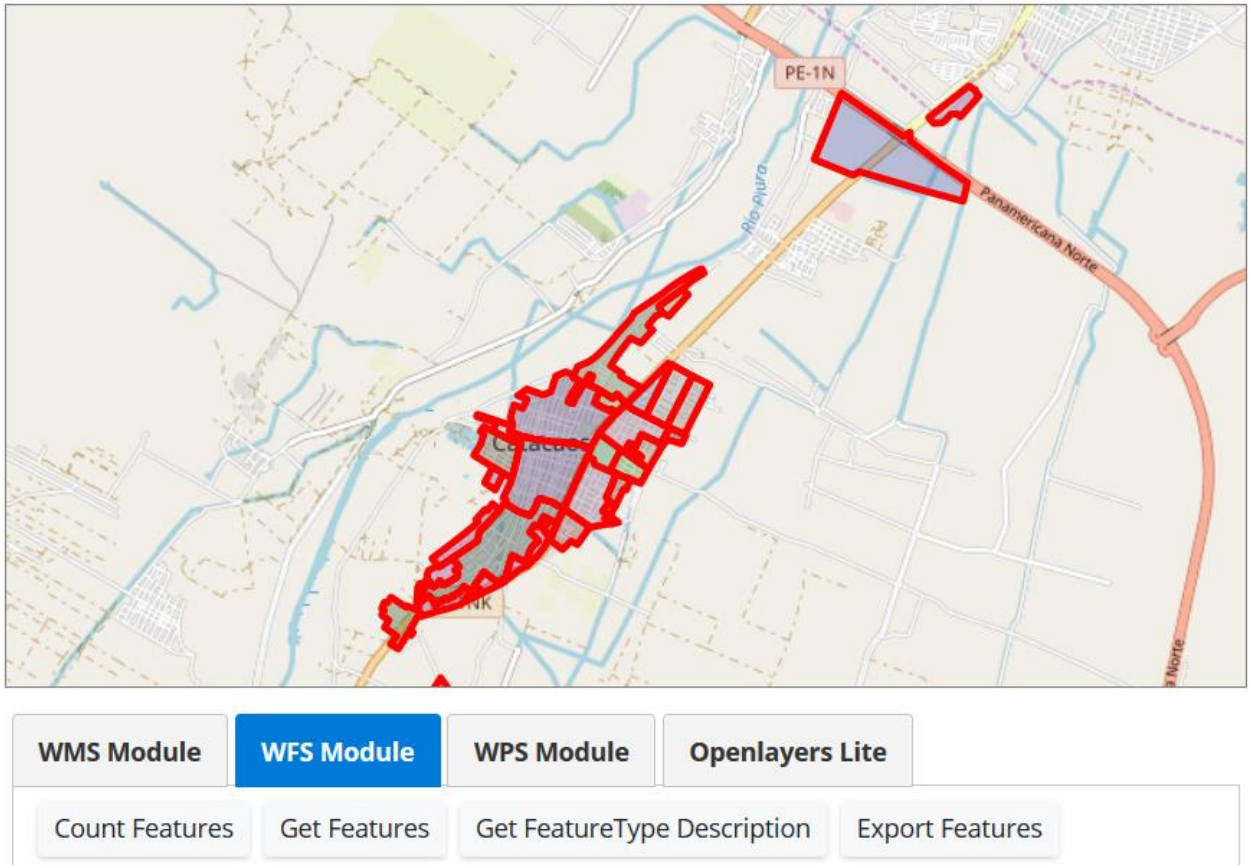


Figura 78. Features correspondientes a las habilitaciones urbanas del distrito de Catacaos, representadas con polígonos sin relleno de borde rojo.

Fuente: Elaboración propia

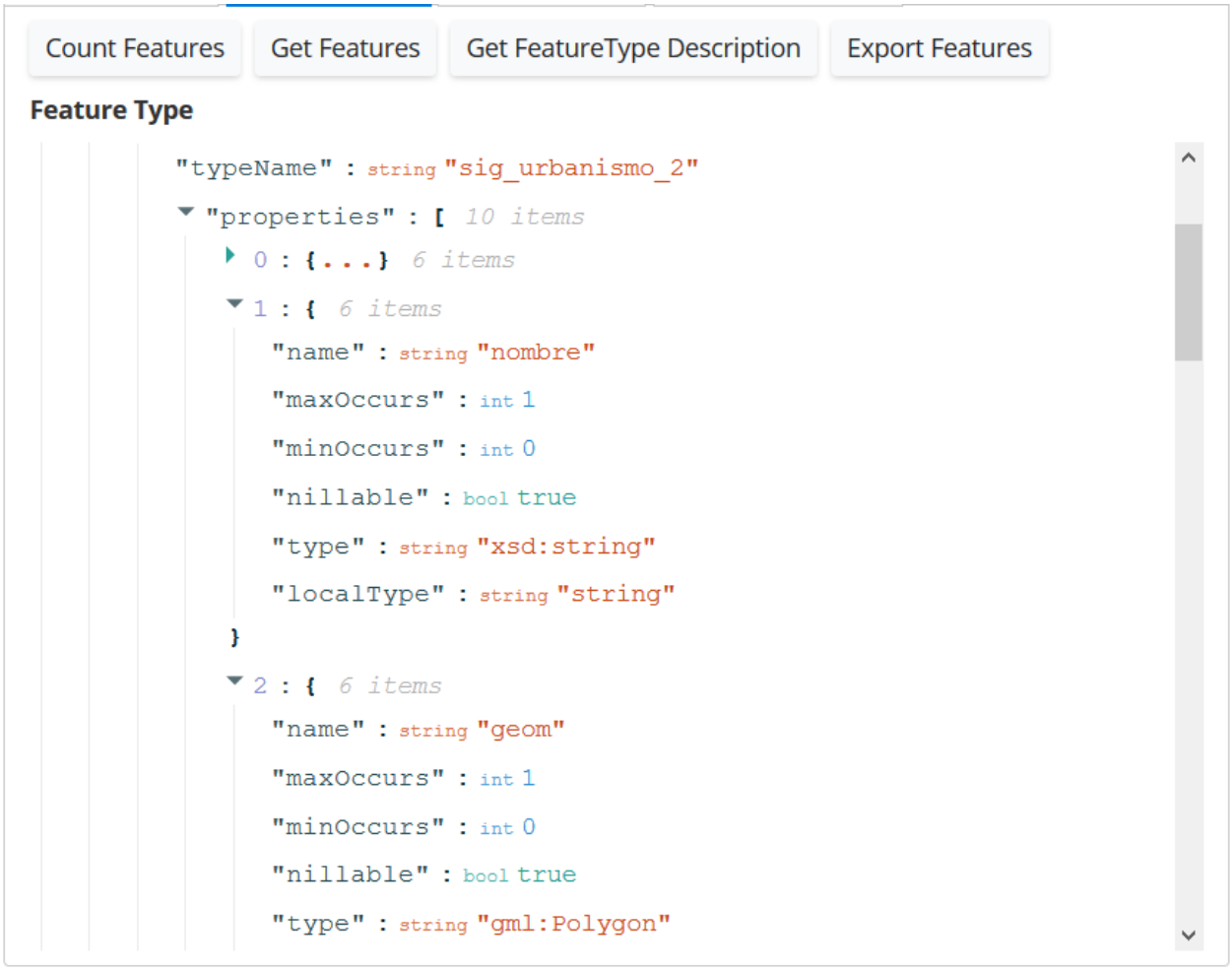


Figura 79. Descripción del tipo de feature de la capa Habilitaciones Urbanas.

Fuente: Elaboración propia

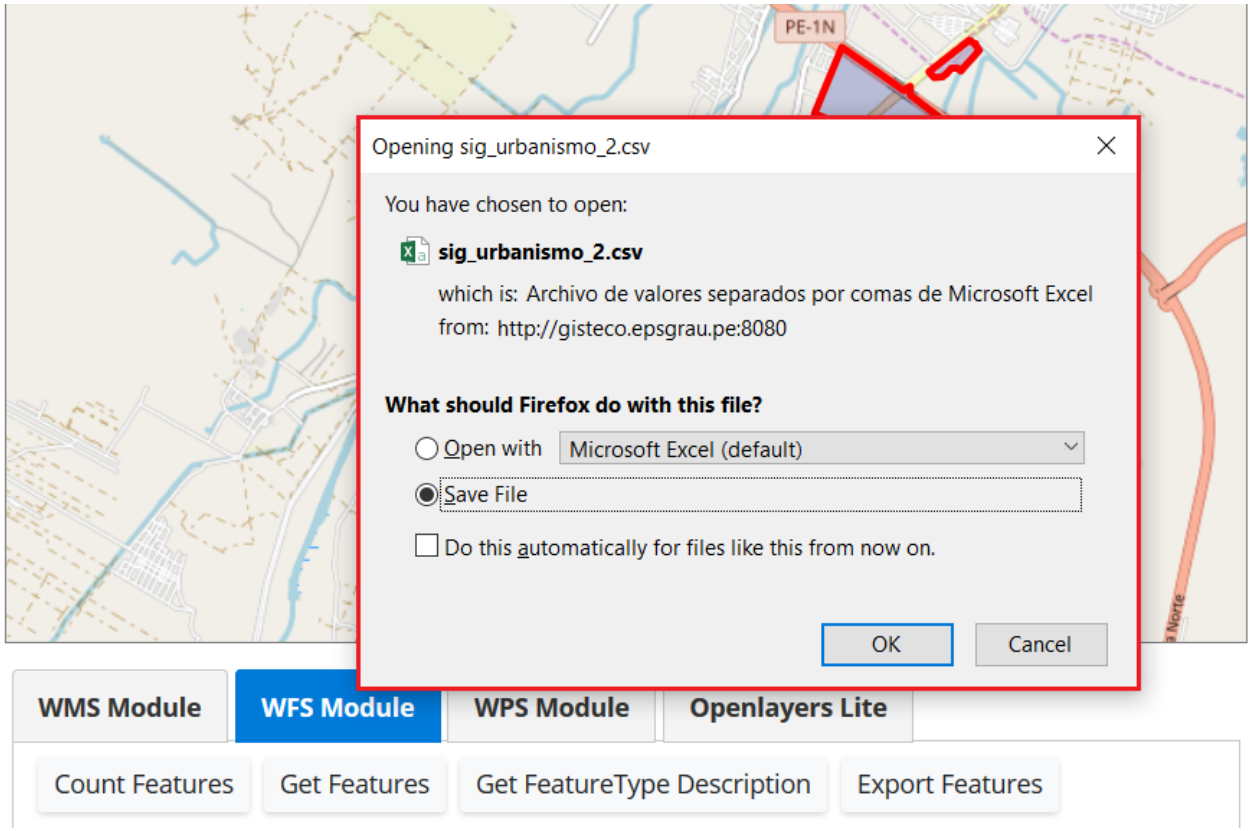
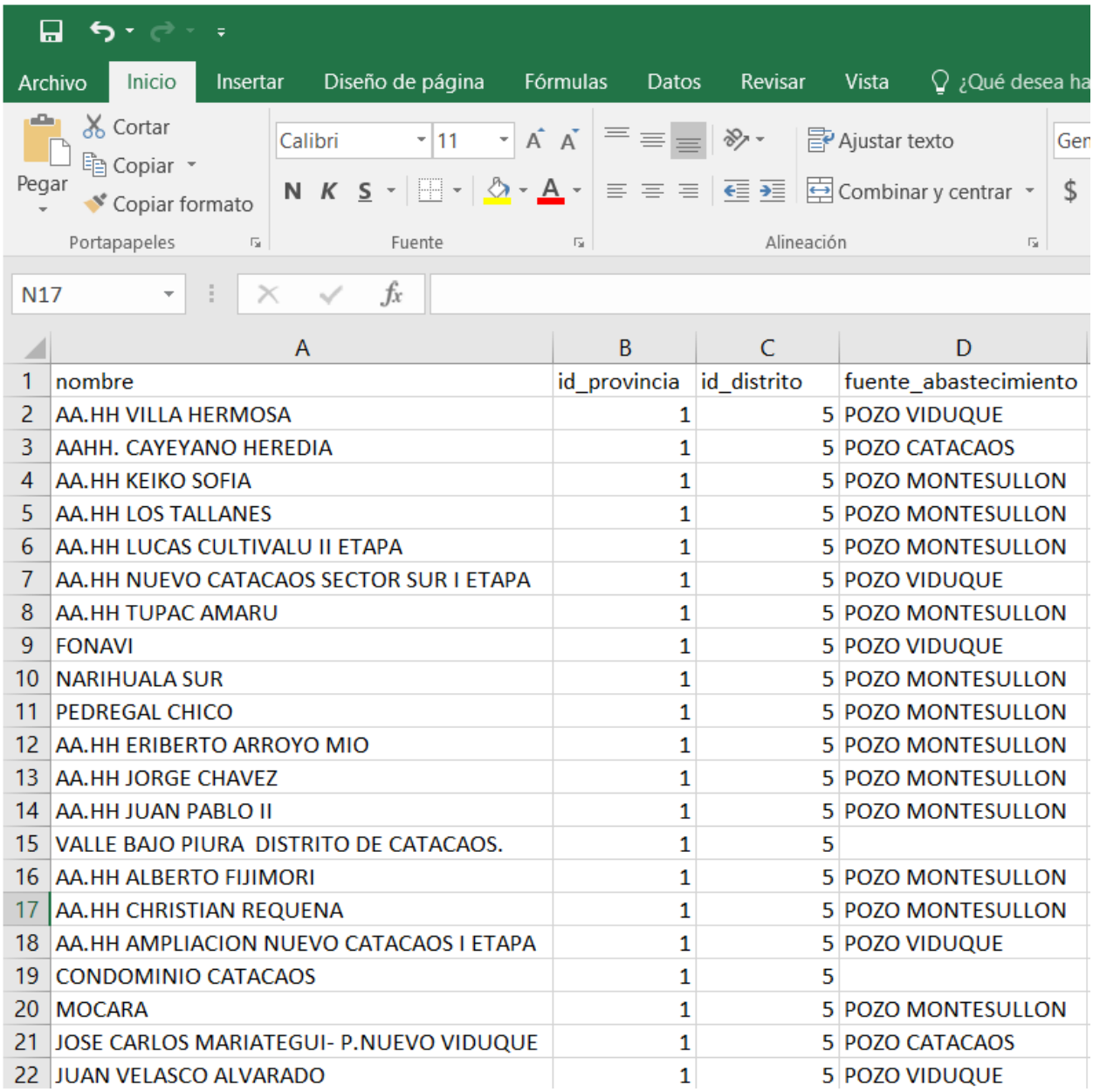


Figura 80. Ventana emergente para la descarga de los features de la capa Habilitaciones Urbanas en formato CSV.

Fuente: Elaboración propia



	A	B	C	D
1	nombre	id_provincia	id_distrito	fuente_abastecimiento
2	AA.HH VILLA HERMOSA	1	5	POZO VIDUQUE
3	AAHH. CAYEYANO HEREDIA	1	5	POZO CATACAOS
4	AA.HH KEIKO SOFIA	1	5	POZO MONTESULLON
5	AA.HH LOS TALLANES	1	5	POZO MONTESULLON
6	AA.HH LUCAS CULTIVALU II ETAPA	1	5	POZO MONTESULLON
7	AA.HH NUEVO CATACAOS SECTOR SUR I ETAPA	1	5	POZO VIDUQUE
8	AA.HH TUPAC AMARU	1	5	POZO MONTESULLON
9	FONAVI	1	5	POZO VIDUQUE
10	NARIHUALA SUR	1	5	POZO MONTESULLON
11	PEDREGAL CHICO	1	5	POZO MONTESULLON
12	AA.HH ERIBERTO ARROYO MIO	1	5	POZO MONTESULLON
13	AA.HH JORGE CHAVEZ	1	5	POZO MONTESULLON
14	AA.HH JUAN PABLO II	1	5	POZO MONTESULLON
15	VALLE BAJO PIURA DISTRITO DE CATACAOS.	1	5	
16	AA.HH ALBERTO FIJIMORI	1	5	POZO MONTESULLON
17	AA.HH CHRISTIAN REQUENA	1	5	POZO MONTESULLON
18	AA.HH AMPLIACION NUEVO CATACAOS I ETAPA	1	5	POZO VIDUQUE
19	CONDOMINIO CATACAOS	1	5	
20	MOCARA	1	5	POZO MONTESULLON
21	JOSE CARLOS MARIATEGUI- P.NUEVO VIDUQUE	1	5	POZO CATACAOS
22	JUAN VELASCO ALVARADO	1	5	POZO VIDUQUE

Figura 81. Archivo en formato CSV descargado con información de los features.

Fuente: Elaboración propia

4.1.2.3. Módulo WPS

La UI para la prueba de las funcionalidades del módulo WPS, está constituida de los siguientes elementos:

1. Botón para obtener el bbox.
2. Espacio para mostrar el bbox.
3. Formulario para configurar los parámetros del proceso de agregación.
4. Botón para ejecutar proceso de agregación.
5. Espacio para mostrar los resultados del proceso de agregación.

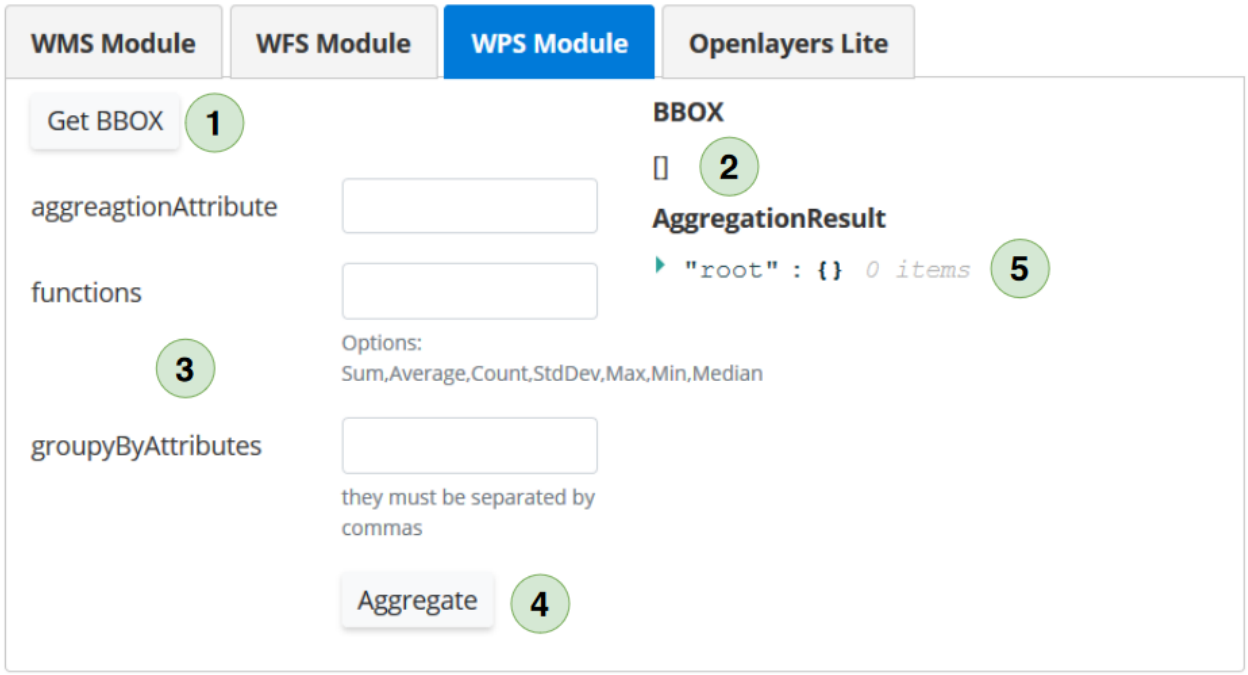


Figura 82. UI para la prueba de funcionalidades del módulo WPS.

Fuente: Elaboración propia

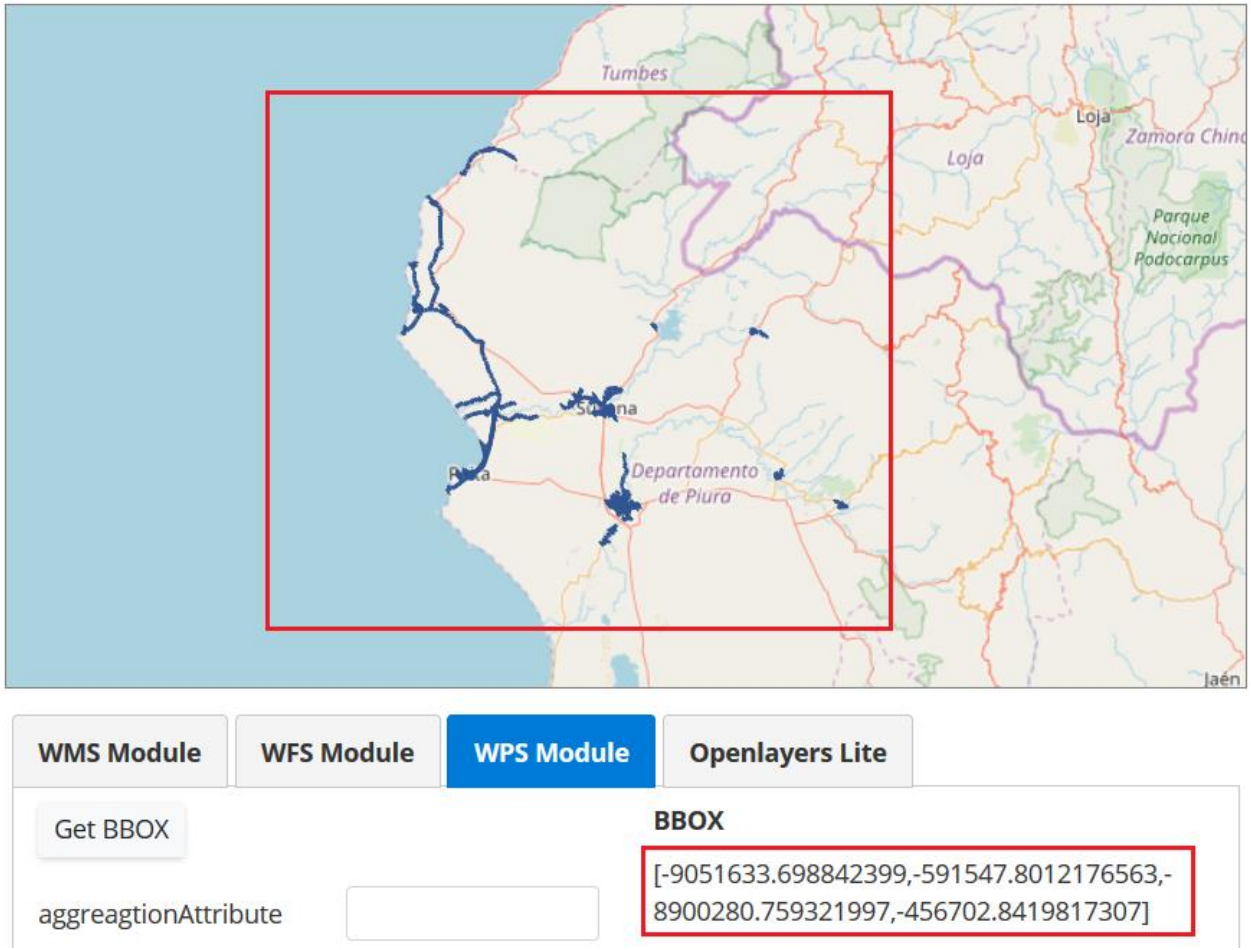


Figura 83. Bounding box de las tuberías de agua administradas por la empresa EPS GRAU S.A.

Fuente: Elaboración propia

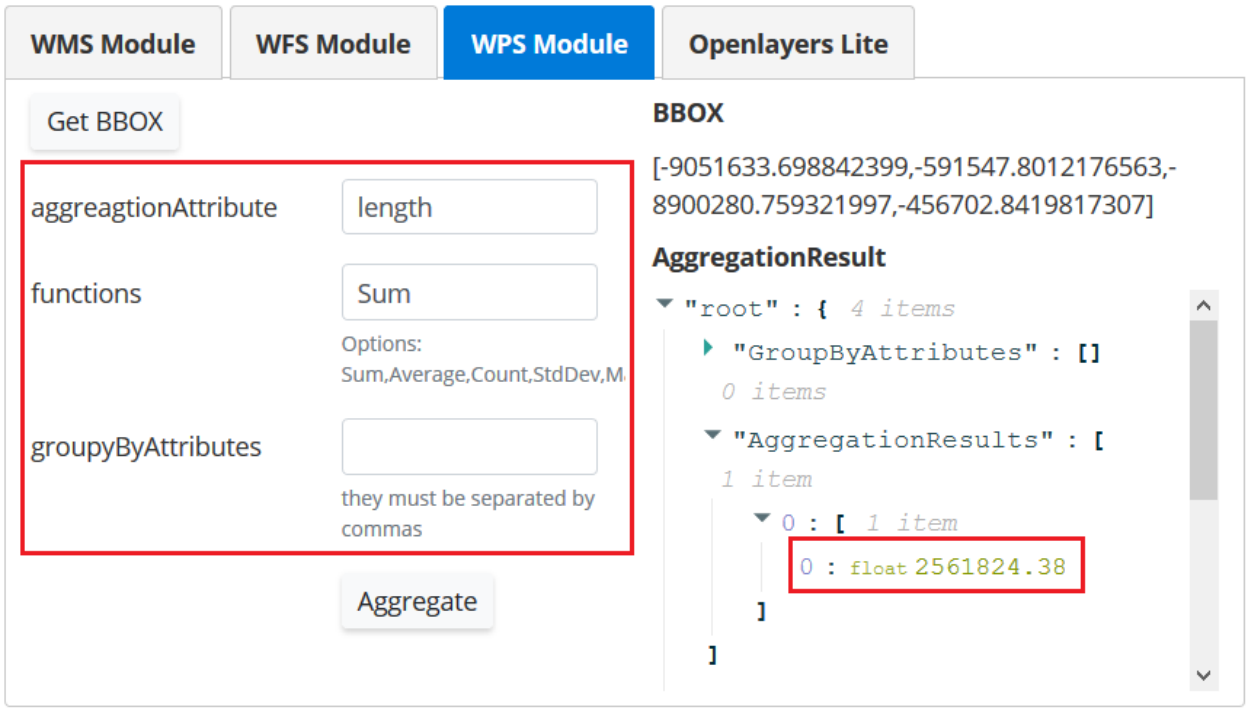


Figura 84. Proceso de agregación ejecutado para calcular el total de metros en tuberías de agua que administra la empresa EPS GRAU S.A.

Fuente: Elaboración propia

4.1.2.4. Módulo OpenlayersLite

La UI para la prueba de las funcionalidades del módulo WPS, está constituida de los siguientes elementos:

1. Botón para dibujar features.
2. Selector del tipo de geometría de los dibujos.
3. Botón para dejar de dibujar.
4. Botón para cargar features con diferentes estilos (correspondientes a las '**tareas asociadas a la creación de estilos para features vectoriales**' de las pruebas de evaluación).

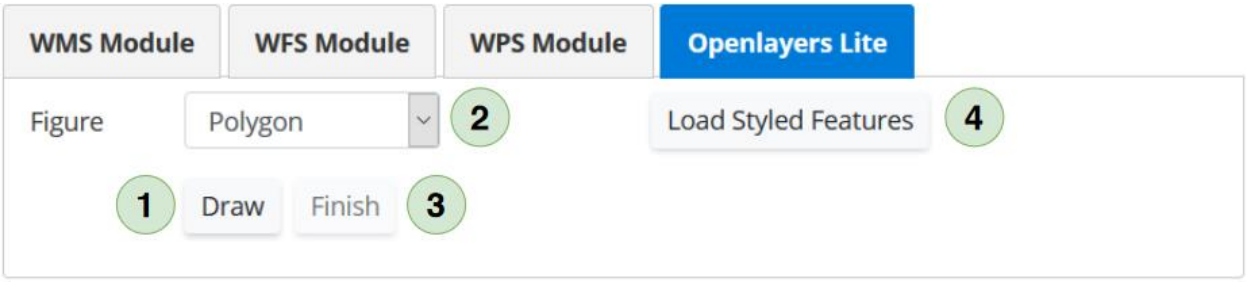


Figura 85. UI para la prueba de funcionalidades del módulo OpenlayersLite.

Fuente: Elaboración propia

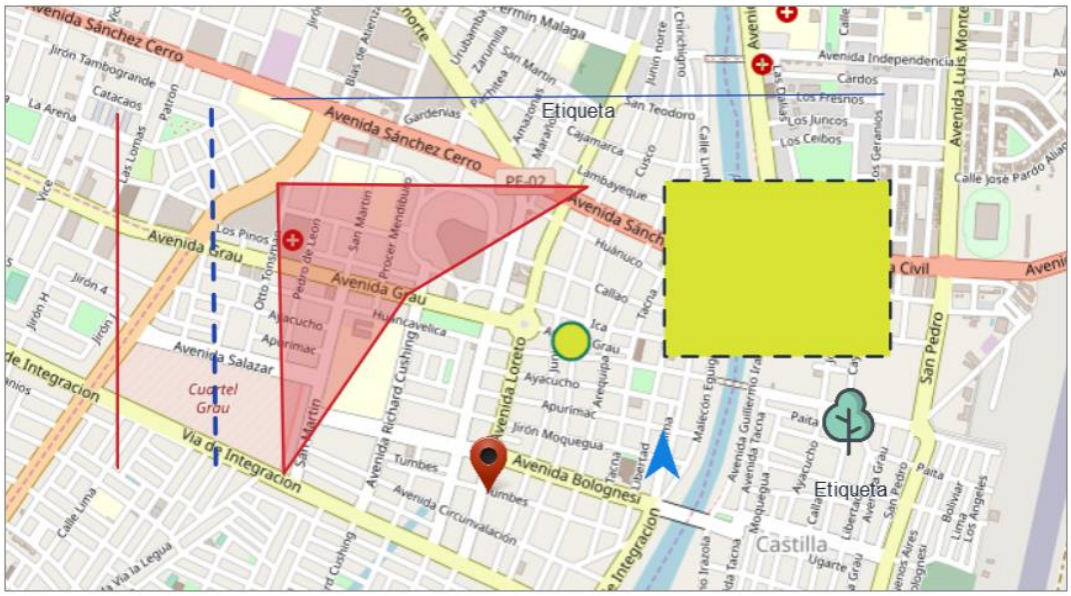


Figura 86. Features vectoriales con estilos creados con easyolmaps.

Fuente: Elaboración propia

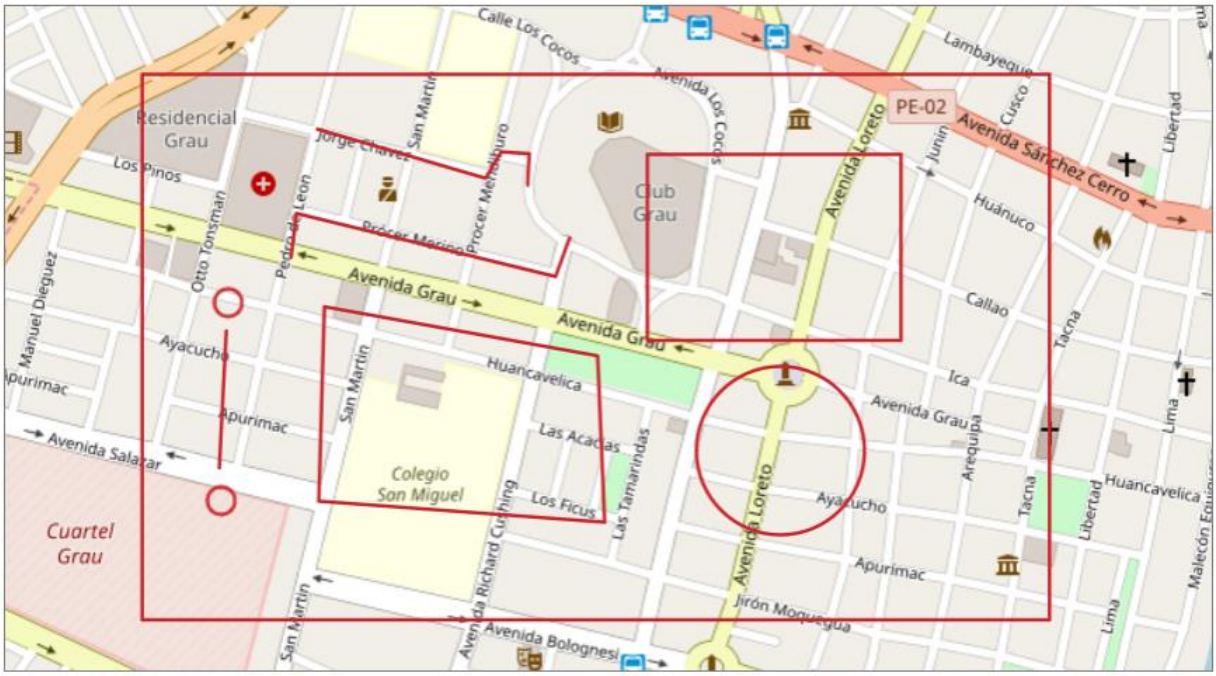


Figura 87. Features vectoriales dibujados con easyolmaps.

Fuente: Elaboración propia

4.1.3. Documentación de la librería

La documentación de la librería se publicó en internet gracias a *GitHub Pages* y se puede acceder a través de la siguiente dirección: https://henryvanner.github.io/easyolmaps/. Al contenido se le dio un formato similar al del sitio de Openlayers, de manera que su lectura sea fácil para los usuarios que ya tienen experiencia con dicha librería.

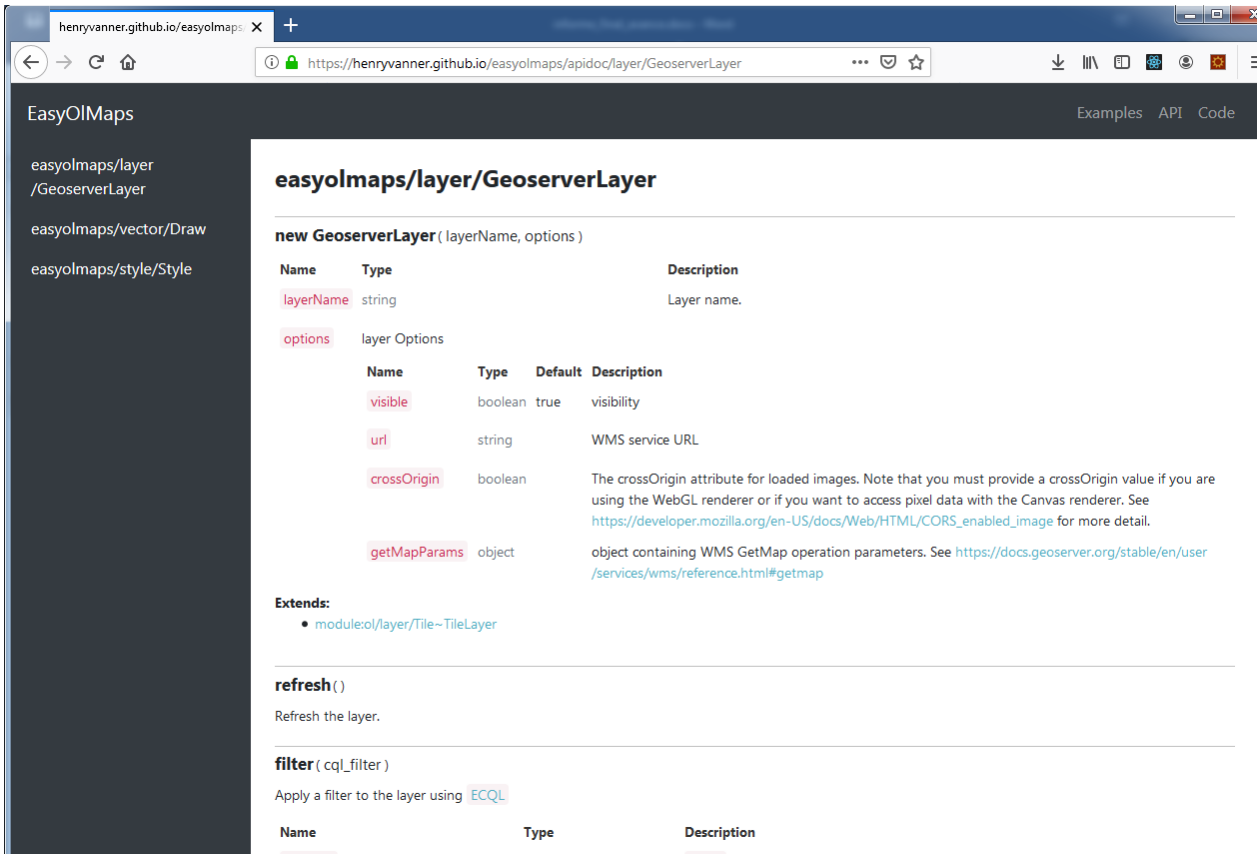


Figura 88. Sitio web de la documentación de easyolmaps.

Fuente: Elaboración propia

4.2.DISCUSIÓN

Sobre la Completitud funcional:

- ✓ El valor de 100% obtenido para CIF indica que la librería cumple con todos los requerimientos funcionales especificados.

Sobre la Coexistencia funcional:

- ✓ El valor de 100% obtenido para COF, señala que, de todas las funciones que fueron implementadas usando la librería y Openlayers, de la aplicación web desarrollada para la evaluación de la librería, todas funcionaron correctamente, demostrando que la librería ciertamente puede trabajar junto con Openlayers sin perjudicar su funcionamiento.

Sobre la Interoperabilidad:

- ✓ El valor de 100% obtenido para PII, indica que todos los tipos de datos de los parámetros de entrada y salida de las funciones de la librería tienen un formato que facilita el intercambio de información y el uso de la información intercambiada con Openlayers, por tanto, se facilita la interoperabilidad entre ambas librerías. En la presente investigación, se dice que un tipo de dato tienen un formato que facilita el intercambio de información si, (1) es un objeto de Openlayers, (2) es un objeto de la librería que está basada en un objeto de Openlayers tal que este lo puede utilizar sin problema alguno, o (3) es un objeto de algunas de las APIs Web; de modo que ambas librerías pueden intercambiar información y utilizar la información intercambiada pues se encuentran en formatos que son soportados por ambas librerías. Esto, contrario al escenario hipotético en que los datos de los parámetros de entrada y salida de las funciones de la librería estuviesen en un formato "extraño" para Openlayers, tal que fuese necesario un mecanismo adicional para poder intercambiar la información y utilizar la información intercambiada.

Sobre la Simplificación del uso de Openlayers:

- ✓ Los datos obtenidos sobre el número de objetos que necesitan la librería y Openlayers para la creación de estilos para features vectoriales (NOL y NOO), muestran que, cuando se usa la librería, el número de objetos que se precisan para la creación de un estilo es 0. Dicho de otro modo, la librería suprime el uso de objetos para la creación de estilos para features vectoriales. Cabe recalcar para prevenir una errónea interpretación de los resultados, que los 'objetos necesarios' referidos, son aquellos que se usan como valores de parámetros para la creación de un

estilo, materializado como un objeto de la clase Style. Por tanto, la creación de este objeto sigue siendo necesaria para la creación de un estilo pues representa el estilo en sí.

- ✓ Con respecto a los resultados obtenidos sobre el número de parámetros que necesitan la librería y Openlayers para la creación de estilos para features vectoriales (NPL y NPO), se pueden observar tres casos: (1) el número de objetos que se necesitan cuando se usa la librería es menor (NPL < NPO), (2) el número de objetos que se necesitan es el mismo para ambas librerías (NPL = NPO) y (3) con una sola ocurrencia, el número de objetos que se necesitan cuando se usa la librería es mayor (NPL > NPO). El caso de las tareas donde NPL fue igual NPO se explica porque para dichas tareas, cada parámetro usado, tanto con la librería como con Openlayers, es imprescindible para que la tarea se cumpla a cabalidad. Mientras que, en el caso de las tareas donde NPL fue menor que NPO, cuando se usó la librería, se pudo prescindir de algunos parámetros pues sus valores por defecto satisfacían los requerimientos, cosa que no se pudo hacer con Openlayers, pues los parámetros correspondientes no tenían valor por defecto asignado, por lo que fue necesaria su asignación explícita. Por otro lado, el caso de la tarea donde NPL fue mayor que NPO, se explica porque con la librería, la transparencia se configura con un parámetro adicional, a diferencia de Openlayers, donde tanto para configurar color y transparencia se hace uso de un único parámetro. Esta forma de configuración de la transparencia con la librería es intencional pues se cree que es más práctico y conveniente que ambas características se configuren por separado, dado que, y tal como se puede constatar revisando su documentación, para configurar la transparencia con Openlayers se debe utilizar ya sea el formato rgba, lo que puede implicar una transformación de formatos si el color no está expresado en un formato diferente, o el formato hexadecimal, que implica una transformación del valor de transparencia, generalmente como un número real entre 0 y 1, a su valor hexadecimal correspondiente.

- ✓ En lo que respecta al número de objetos que se necesitan para dibujar features vectoriales, los datos obtenidos muestran que, cuando se usó la librería, el número de objetos que se precisaron para el dibujo de features fue 0. Si bien el caso fue el mismo cuando se usó Openlayers para dos de las tres tareas, en la tercera tarea si se dio una disminución del número de objetos necesarios. Cabe recalcar, igual que para el caso del número de objetos necesarios para la creación de estilos para features vectoriales, que los ‘objetos necesarios’ referidos son aquellos que se usan como valores de parámetros para la creación de un objeto que sirve para el dibujo de features, materializado como un objeto de la clase Draw. Por tanto, la creación de este objeto sigue siendo necesaria para el dibujo de features.

- ✓ Sobre el número de parámetros que se necesitan para dibujar features vectoriales, los datos obtenidos sugieren, que el número de parámetros que se necesitan utilizando la librería, es el mismo que cuando se usa Openlayers, sin embargo, esto no es cierto para todos los casos, como sucedió en la tarea 2, donde NPL es menor que NPO por una unidad. Esto se explica en el hecho de que, con Openlayers, para poder dibujar features con formas geométricas que no sean las de puntos, líneas o polígonos; como en el caso de la tarea 2 que requiere el dibujo de rectángulos; hace falta la implementación de una función que define la geometría para un feature y la configuración de un parámetro para indicar que dicha función debe ser usada en los dibujos. Caso contrario a cuando se usa la librería, donde no se requiere de la especificación de esta función para ninguna de las formas geométricas que puede dibujar. De modo que usando la librería se pueden empezar a dibujar features con formas geométricas sin mayor especificación de la forma geométrica requerida que el tipo de geometría.

CONCLUSIONES

- ✓ Después de revisar 53 visores de mapas que figuran en el sitio web de la IDEP, de distintas entidades públicas y privadas del país, que corresponden a diferentes áreas de estudio o interés, y con diferentes fines de uso y grados de complejidad, se observó que varios de ellos tienen funcionalidades comunes asociadas a los servicios WMS, WFS y WPS. Luego se especificaron la lista de funcionalidades que serían incluidas en la librería, en base a dichas observaciones y según los lineamientos de esta investigación que dictan que los requerimientos del software deben ser funcionalidades relacionadas a los servicios WMS, WFS y WPS de GeoServer y que el software debe ser compatible con Openlayers.

- ✓ Para el desarrollo de la librería, se requería la creación de un modelo que cumpla con los requerimientos especificados. Para este fin, se hizo uso del lenguaje de modelamiento unificado (UML), con el cual se creó un modelo dividido convenientemente en módulos, cada uno de los cuales modelan un aspecto distinto de la librería que cumple con una parte de los requerimientos, asociados a uno de los servicios de GeoServer, que en conjunto conforma un modelo que satisface todos los requerimientos de la librería establecidos.

- ✓ Se llevó a cabo el desarrollo (codificación) de la librería según el modelo creado. Para ello, se creó un repositorio en *github* bajo el nombre de *easyolmaps*, y se utilizó el sistema de control de versiones *git* para llevar un control de las versiones del código.

- ✓ Conjuntamente con el desarrollo de la librería, se empezó el desarrollo de su documentación en versión web, a la cual se le dio un formato similar al del sitio web de Openlayers, de manera que su lectura sea fácil para los usuarios que ya tienen experiencia con dicha librería. La publicación del sitio web fue posible de manera gratuita gracias al servicio *Github Pages*.

- ✓ Los resultados obtenidos de la evaluación de la librería sobre las características y sub-características de la calidad evaluadas que son: funcionalidad (completitud funcional), compatibilidad (coexistencia e interoperabilidad con Openlayers) y usabilidad (simplificación del uso de Openlayers, en lo que respecta a las funcionalidades de creación de estilos para features vectoriales y dibujo de features vectoriales); muestran que: (1) la librería cumple con todos los requerimientos funcionales especificados (CIF = 100%), por tanto, implementa funcionalidades relacionadas a los servicios WMS, WFS y WPS de GeoServer; (2) la librería es compatible con Openlayers (COF = 100% y PII = 100%); y (3) la librería simplifica el uso de Openlayers, precisando que en las pruebas realizadas, en relación con la creación de estilos, se observó que con la librería se disminuyó a 0 el número de objetos necesarios, y que el número de parámetros

necesarios se redujo en un promedio de 1.833, con un valor mínimo de -1 y un máximo de 4 parámetros; en lo que respecta al dibujo de features, se observó que, aunque en dos de tres tareas, ambos la librería y Openlayers no precisaron de ningún objeto, en la última tarea, mientras que Openlayers necesitó de un objeto, la librería mantuvo su tendencia y no requirió de parámetro alguno, y que la librería suprime la necesidad de especificar una función que determina el tipo de geometría cuando se quiere dibujar features con formas geométricas que no son puntos, línea o polígonos; lo que representa 1 parámetro menos y 1 función menos a implementar. Así, lo anterior se puede subsumir, en que la librería desarrollada en el presente trabajo de investigación, cumple con su propósito, el de ser una librería basada en Openlayers, para aplicaciones web de mapas, que implemente funcionalidades relacionadas a los servicios WMS, WFS y WPS de GeoServer.

RECOMENDACIONES

Para los investigadores o profesionales que estén interesados en contribuir con la mejora de la librería, se recomienda:

- ✓ Realizar el análisis y desarrollo necesario para que la librería pueda ser usada en aplicaciones web de mapas que usen MapServer, dado que, en otros trabajos de investigación que preceden a este, se constató que este servidor junto con GeoServer, son los servidores de mapas OpenSource que obtuvieron los mejores resultados en pruebas de rendimiento sometidas a sus respectivos servicios WMS y WFS.

- ✓ Ampliar la cobertura de la librería de los estándares WMS, WFS y WPS, incluyendo funciones para la ejecución de operaciones tales como: GetPropertyValue, LockFeature, Transaction y GetFeatueWithLock de WFS, y para la ejecución de otros procesos disponibles a través de los servicios WPS, o la creación de un mecanismo general para la ejecución de procesos espaciales disponibles a través de un servicio WPS; a la vez que actualizar, de ser necesario, las funciones existentes a fin de asegurar el funcionamiento de la librería en el tiempo.

Para los investigadores o profesionales que estén interesados en desarrollar algún producto de software a partir de la librería, se recomienda:

- ✓ Desarrollar junto con Openlayers, un sistema web de información geográfica de propósito general, de modo que pueda servir para la realización de futuros trabajos, estudios o investigaciones, y para impulsar el uso, mantenimiento y mejora de estas herramientas OpenSource.

Para los investigadores que estén interesados en realizar un aporte o mejora al presente trabajo de investigación y otros semejantes, se recomienda:

- ✓ Desarrollar un método para la evaluación de calidad del software más actualizado y completo que el empleado en este trabajo; incluyendo en el informe, un mayor número de métricas para evaluar las características y sub-características de la calidad, y una mejor explicación del modo de aplicación del método mismo y de los procedimientos para obtener los valores de las métricas, a fin de contribuir con el desarrollo de un mayor número de productos de software de calidad.

REFERENCIAS BIBLIOGRÁFICAS

Babel. (2019). *Babel*. Recuperado de https://babeljs.io/

Baldeón Villanes, E. J. (2015). *Método para la evaluación de calidad de software basado en ISO/IEC 25000.* Lima.

Eckel, B. (2000). *Thinking in C++*. New Jersey: Prentice Hall.

EPS GRAU, E. P. (2019). *GISTECO*. Recuperado de http://gisteco.epsgrau.pe

Facebook Inc. (2019). *React*. Recuperado de https://reactjs.org/

Fernandes, A. I. (2012). *Estudo comparativo entre Interfaces de Programação de Aplicações de mapas.* Lisboa.

Git. (2019). *Git*. Recuperado de https://git-scm.com/

Github. (2019). *Github*. Recuperado de https://github.com/

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la Investigación.* México: Mc Graw Hill Education.

IDEP, I. d. (2019). *Visor de Mapas del Perú*. Recuperado de http://mapas.geoidep.gob.pe/mapasperu/

IDEP, I. d. (2019). *Visores de Mapas*. Recuperado de https://www.geoidep.gob.pe

INDECI, I. N. (2019). *Rutas de Evacuación y Zonas Seguras*. Recuperado de http://sinpad.indeci.gob.pe/Evacuacion/index.html

Internet Engineering Task Force. (2019). *The GeoJSON Format*. Recuperado de https://tools.ietf.org/html/rfc7946

Khan Academy. (2019). *What's a JS library?* Recuperado de https://www.khanacademy.org/computing/computer-programming/html-css-js/using-js-libraries-in-your-webpage/a/whats-a-js-library

MDN, M. D. (2019). *JavaScript*. Recuperado de https://developer.mozilla.org/en-US/docs/Web/JavaScript

Microsoft. (2019). *¿Qué es un archivo DLL?* Recuperado de https://support.microsoft.com/es-pe/help/815065/what-is-a-dll

Microsoft. (2019). *Well Known Text Module*. Recuperado de https://docs.microsoft.com/en-us/bingmaps/v8-web-control/modules/well-known-text-module

MINAGRI, M. d. (2019). *Sistema Catastral Rural para Predios Rurales Individuales*. Recuperado de http://georural.minagri.gob.pe/sicar/

MINAM, M. d. (2019). *Geoservidor*. Recuperado de http://geoservidorperu.minam.gob.pe/geominam

MINAM, M. d. (2019). *Sistema de Información para Zonas Marino Costera*. Recuperado de http://geoservidorperu.minam.gob.pe/spincam

MININTER, M. d. (2019). *Ubica tu comisaría*. Recuperado de https://www.mininter.gob.pe/ubica-tu-comisaria

Mitchell, T. (2005). *Web Mapping Illustrated.* O'Really.

MTC, M. d. (2019). *Visor de Emergencias Viales*. Recuperado de http://sigeo.mtc.gob.pe/visorcoe01mtc2017/

Node.js Foundation. (2019). *NodeJS*. Recuperado de https://nodejs.org/en/

NPM. (2019). *npm*. Recuperado de https://www.npmjs.com/

Object Managment Group. (2019). *Unified Modeling Language.* Recuperado de https://www.omg.org/spec/UML/2.5/PDF

OGC, O. G. (2002, 01 16). *Web Map Service Implementation Specification.* Recuperado de http://portal.opengeospatial.org/files/?artifact_id=1081&version=1&format=pdf

OGC, O. G. (2005, 05 03). *Web Feature Service Implementation Specification.*

OGC, O. G. (2007, 06 08). *OpenGIS Web Processing Service.* Recuperado de http://portal.opengeospatial.org/files/?artifact_id=24151

OGC, O. G. (2007, 06 29). *Styled Layer Descriptor profile of the Web Map Service Implementation Specification.* Recuperado de http://www.opengeospatial.org/docs/is

OGC, O. G. (2009, 01 15). *Features.* Recuperado de https://portal.opengeospatial.org/files/?artifact_id=29536

Olaya, V. (2014). *Sístemas de Información Geográfica.* Recuperado de https://github.com/volaya/libro-sig/releases/download/v2.0/Libro_SIG.pdf

Openlayers. (2019). *Openlayers*. Recuperado de https://openlayers.org/

OSGeo, O. S. (2019). *Geoserver*. Recuperado de http://geoserver.org/

OSINFOR, O. d. (2019). *SISFOR*. Recuperado de https://sisfor.osinfor.gob.pe/visor/

Santiago, A. (2015). *The book of Openlayers 3.* Leanpub.

SENAMHI, S. N. (2019). *IDESEP - SENAMHI*. Recuperado de http://idesep.senamhi.gob.pe/geovisoridesep/

SERFOR, S. N. (2019). *GEOSERFOR*. Recuperado de http://geo.serfor.gob.pe/visor/

W. Kernighan, B., & M. Ritchie, D. (1988). *The C Programming Language.* New Jersey: Prentice Hall.

ANEXOS

ANEXO 1. REQUISITOS DE EVALUACIÓN DE CALIDAD

Requisitos de Evaluación de Calidad

1. Información de Identificación

1.1. Identificación del Evaluador

Nombre		**Formación Académica**	
Ocupación			

1.2. Identificación del Reporte de Evaluación

Reporte N°		**Fecha**	

2. Dominio de la aplicación

Librería para el desarrollo de aplicaciones web de mapas, que requieran de funciones relacionadas con los servicios WMS, WFS y WPS de GeoServer.

3. Propósito de la evaluación

Evaluar de calidad del software (producto terminado e implementado). Verificar que cumpla con los requisitos funcionales y no funcionales definidos.

4. Lista de requerimientos de calidad

Se requiere la evaluación de las siguientes características:

Funcionalidad	
Completitud funcional	Grado en que el conjunto de funciones cubre todas las tareas y objetivos (requerimientos) especificados.
Compatibilidad	
Coexistencia	Grado en que un producto puede llevar a cabo sus funciones de manera eficiente mientras comparten un entorno y recursos con otros productos, sin impacto perjudicial en los otros productos.
Interoperabilidad	Grado en el cual dos o más sistemas, productos o componentes pueden intercambiar información y utilizar la información que se ha intercambiado.
Usabilidad	
Simplificación del uso de Openlayers.	Grado en que la librería simplifica el uso de las funcionalidades de Openlayers.

5. Productos a ser incluidos en la evaluación

La librería, que está disponible a través del sistema de gestión de paquetes npm bajo el nombre *easyolmaps*.

Henry Vanner Aquino Vegas
DNI. 74243367
Solicitante

Firma del evaluador

ANEXO 2. ESPECIFICACIÓNES DE LA EVALUACIÓN DE CALIDAD

Especificaciones de la Evaluación de Calidad

1. Información de Identificación

1.1. Identificación del Evaluador

Nombre		**Formación Académica**	
Ocupación			

1.2. Identificación del Reporte de Evaluación

Reporte N°		**Fecha**	

2. Alcance de la evaluación

Realizar la evaluación de la calidad de la librería. Verificar y medir el cumplimiento de los requerimientos funcionales y no funcionales.

3. Componentes necesarios

Para llevar a cabo la evaluación, los siguientes elementos e información se pondrán a disposición:

- Librería.
- Requisitos funcionales y no funcionales.
- Documentación técnica y de usuario de la librería.

4. Métricas de calidad y criterios de decisión

ID	Nombre	Descripción	Característica evaluada	Función
CIF	Cobertura de implementación funcional	¿Qué tan completa es la implementación de acuerdo a las especificaciones de los requerimientos?	Completitud funcional	X = 1 – A/B, donde: A = Número de funciones que falta o están incorrectamente implementadas, que han sido detectadas en la evaluación. B = Número de funciones establecidas en la especificación de requerimientos.

Nota: Una función que falta o está incorrectamente implementada puede ser:

a) Función que no funciona según lo especificado en los manuales de usuario, especificación de requerimientos o especificaciones de diseño.
b) Función que no proporciona un resultado razonable y aceptable para lograr su objetivo.

COF	Coexistencia funcional	¿Qué parte de las funciones implementadas se ejecutan sin errores cuándo son usadas junto con Openlayers?	Coexistencia	X = A/B, dónde: A = Número de funciones que se ejecutaron correctamente y sin errores. B = Número de funciones que fueron implementadas con la librería y con Openlayers.
PII	Pertinencia para el intercambio de información	¿Qué parte de los tipos de datos de los parámetros de entrada y salida de las funciones de la librería tienen un formato que facilita el intercambio de información con Openlayers?	Interoperabilidad	X = A/B, dónde: A = Número de tipos de datos de parámetros que tienen un formato que facilita el intercambio de información con Openlayers. B = Número total de tipos de datos de los parámetros.
Nota: Se evalúan solo los tipos de datos que no son tipos de datos primitivos ni objetos JSON. Un tipo de dato de un parámetro de entrada o salida de una función de la librería tienen un formato que facilita el intercambio de información si: a) Es un objeto de Openlayers. b) Es un objeto de la librería que está basado en un objeto Openlayers, tal que Openlayers lo puede utilizar sin producirse error alguno. Por ejemplo: los objetos de una clase que extiende de una clase de Openlayers. c) Es un objeto de algunas de las APIs Web.				
SNO	Simplificación del número de objetos	¿En cuánto disminuyó el número de objetos necesarios para usar una funcionalidad?	Simplificación del uso de Openlayers	X = B - A, dónde: A = Número de objetos necesarios para usar la funcionalidad con la librería. B = Número de objetos necesarios para usar la funcionalidad con Openlayers.
SNP	Simplificación del número de parámetros	¿En cuánto disminuyó el número de parámetros necesarios para usar una funcionalidad?	Simplificación del uso de Openlayers	X = B - A, dónde: A = Número de parámetros necesarios para usar la funcionalidad con la librería. B = Número de parámetros necesarios para usar la funcionalidad con Openlayers.

5. Métodos de evaluación

- Revisión de la especificación de los requerimientos de la librería.
- Revisión de documentación técnica y de usuario.
- Pruebas de la librería para la realización de tareas específicas.

Henry Vanner Aquino Vegas DNI. 74243367 **Solicitante**	**Firma del evaluador**

ANEXO 3. DISEÑO DE LA EVALUACIÓN DE CALIDAD

Diseño de la Evaluación de Calidad

1. Información de Identificación

1.1. Identificación del Evaluador

Nombre		**Formación Académica**	
Especialidad			

1.2. Identificación del Reporte de Evaluación

Reporte N°		**Fecha**	

2. Plan de trabajo

La evaluación tendrá una duración máxima de 4 días y comprenderá las siguientes actividades.

Actividad	Encargado	Día 1	Día 2	Día 3	Día 4
Revisión de la librería	Evaluador				
Revisión de la información • **Requerimientos de la librería** • **Documentación técnica y de usuario**	Evaluador				
Especificación de las pruebas	Evaluador				
Diseño de las pruebas	Evaluador				
Configuración del entorno de pruebas	Evaluador				
Ejecución de las pruebas	Evaluador				
Ejecución de las mediciones	Evaluador				
Interpretación de los datos	Evaluador				
Elaboración del borrador del informe final de evaluación	Evaluador				
Revisión del borrador del informe	Solicitante y Evaluador				
Elaboración del informe final de evaluación	Evaluador				
Entrega del informe final	Evaluador				

Henry Vanner Aquino Vegas
DNI. 74243367
Solicitante

Firma del evaluador

ANEXO 4. PRUEBAS PARA LA EVALUACIÓN DE LA CALIDAD

Pruebas para la evaluación de la calidad

1. Información de Identificación

Característica Evaluada	Completitud funcional, Coexistencia funcional		
Evaluador			
Documento N°		**Fecha**	

2. Objetivo

Evaluar la completitud funcional de la librería y la coexistencia funcional cuando se usa junto con Openlayers en el desarrollo de una aplicación web.

3. Metodología

El evaluador deberá desarrollar una aplicación web que cumpla con los requerimientos especificados en el **apartado 5**, usando la librería y Openlayers.

4. Entorno de pruebas

Las pruebas se llevarán a cabo en un entorno adecuado para el desarrollo de una aplicación web, usando las siguientes tecnologías:

Item	Versión
Node + npm	10.15.3
JavaScript	ES6 y superiores
GeoServer	2.10.0
Openlayers	5.3.2
Easyolmaps	0.0.8

Tabla 1. Tecnologías usadas para las pruebas.

5. Requerimientos de la aplicación web

1. Mostrar una capa ráster del servidor de mapas.
2. Opción para cambiar el estilo de la capa.
3. Opción para ver la leyenda de la capa según el estilo asignado.
4. Opción para seleccionar un elemento (feature) de la capa y mostrar su información.
5. Opción para filtra el contenido de la capa usando una expresión ECQL.
6. Opción para contar los elementos de la capa.
7. Opción para obtener los features de la capa y mostrarlos en el mapa.
8. Opción para obtener la descripción del tipo de feature de la capa.
9. Opción para exportar los features de la capa en un archivo con formato CSV.
10. Opción para obtener el bounding box de la capa y hacer zoom sobre él.
11. Formulario para ejecutar funciones de agregación sobre los features de la capa.
12. Opción para dibujar features, que usen un estilo creado con la librería.
13. Opción para filtrar los elementos de la capa que estén dentro del área de un polígono dibujado por el usuario.

Firma del evaluador

Pruebas para la evaluación de la calidad

1. Información de Identificación

Característica Evaluada	Simplificación del uso de Openlayers		
Evaluador			
Documento N°		**Fecha**	

2. Objetivo

Medir la simplificación del uso de Openlayers para la realización de las siguientes tareas:

- Creación de estilos para features vectoriales.
- Dibujo de features.

3. Metodología

El evaluador deberá implementar en una aplicación web, cada una de las tareas especificadas en el **apartado Nº 5**, usando la libería y Openlayers por separado. Es decir, por cada tarea deberá realizar dos implementaciones, una usando la librería y otra usando Openlayers.

4. Entorno de pruebas

Las pruebas se llevarán a cabo en un entorno adecuado para el desarrollo de una aplicación web, usando las siguientes tecnologías:

Item	Versión
Node + npm	10.15.3
JavaScript	ES6 y superiores
GeoServer	2.10.0
Openlayers	5.3.2
Easyolmaps	0.0.8

Tabla 1. Tecnologías usadas para las pruebas.

5. Tareas

5.1. Tareas asociadas a la creación de estilos para features vectoriales

1. Crear un estilo para una línea, con las siguientes características:
    a. Color de borde rojo (#ce232b).
    b. Ancho de borde de 2 píxeles.
2. Crear un estilo para una línea, con las siguientes características:
    a. Color de borde azul (#1f43ae).
    b. Ancho de borde de 3 píxeles.
    c. Estilo de línea discontinua.
3. Crear un estilo para un polígono, con las siguientes características.
    a. Color de borde rojo (#ce232b).

    b. Ancho de borde de 2 píxeles.
    c. Color de relleno de rojo (#ce232b).
    d. Transparencia del relleno con un factor de 0.35.
4. Crear un estilo para un polígono, con las siguientes características.
    a. Color de borde negro (#212f3d).
    b. Ancho de borde de 2 píxeles.
    c. Estilo de borde de línea discontinua.
    d. Color de relleno amarillo (#d8e333).
    e. Estilo de esquinas redondeado.
5. Crear un estilo para un punto, con las siguientes características.
    a. Color de borde verde (#0c6f58).
    b. Ancho de borde de 2 píxeles.
    c. Radio de 12 píxeles.
    d. Color de relleno amarillo (#d8e333).
6. Crear un estilo para un punto, con las siguientes características.
    a. Que muestre la imagen **Marker** (ver tabla 1)
    b. Escalada por un factor de 0.15
    c. Anclada a la mitad de la parte inferior. (anchor = [0.5,1]).
7. Crear un estilo para un punto, con las siguientes características.
    a. Que muestre la imagen **Arrow** (ver tabla 1)
    b. Escalada por un factor de 0.25
    c. Rotación de 45 grados en sentido antihorario.
8. Crear un estilo para para una línea, que tenga las siguientes características.
    a. Color de borde azul (#1f43ae).
    b. Con una etiqueta que tenga un estilo de letra según las siguientes especificaciones:
        i. Tamaño de letra de 13 pixeles.
        ii. Color negro (#212f3d).
        iii. Borde de color blanco (#ffffff).
        iv. Con un desplazamiento vertical positivo de 10 píxeles.
9. Crear un estilo para un punto, que tenga las siguientes características.
    a. Que muestre la imagen **Tree** (ver tabla 1).
    b. Escalada por un factor de 0.2
    c. Anclada a la mitad de la parte inferior. (anchor = [0.5,1]).
    d. Con una etiqueta que tenga un estilo de letra según las siguientes especificaciones:
        i. Tamaño de letra de 13 píxeles.
        ii. Color negro (#212f3d).
        iii. Borde de color blanco (#ffffff).
        iv. Con un desplazamiento vertical positivo de 20 píxeles.

5.2. Tareas asociadas al dibujo de features

1. Crear un objeto para dibujar polígonos.
2. Crear un objeto para dibujar rectángulos.
3. Crear un objeto para dibujar líneas, con funcionalidad de snapping incluida.

6. Tablas

Imagen	URL
Marker	https://i.dlpng.com/static/png/268757_preview.png
Arrow	https://image.flaticon.com/icons/png/128/149/149049.png
Tree	https://img.icons8.com/cotton/2x/tree.png

Tabla 2. Imágenes.

Firma del evaluador

ANEXO 5. GUÍA DE OBSERVACIÓN PARA EVALUAR LA COMPLETITUD FUNCIONAL

Guía de observación N° 1

1. Información de Identificación

Característica Evaluada	Completitud funcional		
Evaluador			
Documento N°		**Fecha**	

2. Objetivo

Verificar el cumplimiento de los requerimientos funcionales.

3. Instrucciones

En la columna "Cumple", escriba "SI" si la librería tiene una función que cumple con el requerimiento y si dicha función está correctamente implementada, o escriba "NO" si la librería no tiene una función que cumpla con el requerimiento o está incorrectamente implementada..

4. Requerimientos

N°	Requerimiento	Cumple	Observaciones
1	Se puede crear una capa WMS, en base a una capa de servidor WMS de GeoServer.		
2	Se puede cambiar el estilo de la capa.		
3	Se puede aplicar un filtro ECQL a la capa.		
4	Se puede obtener features de la capa en una ubicación dada.		
5	Se puede obtener la leyenda de la capa.		
6	Se puede obtener features de la capa.		
7	Permite la exportación de los features de la capa.		
8	Se puede obtener el número de elementos de la capa.		
9	Se puede obtener la descripción del tipo de feature de la capa.		
10	Se puede obtener el *bounding box* de la capa.		
11	Se puede ejecutar funciones de agregación a los features.		

Tabla 1. Requerimientos funcionales de Easyolmaps.

Firma del evaluador

ANEXO 6. GUÍA DE OBSERVACIÓN PARA EVALUAR LA COEXISTENCIA FUNCIONAL

Guía de observación N° 2

1. Información de Identificación

Característica Evaluada	Coexistencia funcional				
Evaluador					
Documento N°		**Doc. Referencia**		**Fecha**	

2. Objetivo

Verificar en la aplicación web (referida en el documento de referencia), el correcto funcionamiento de las opciones para cuya implementación se requiere del uso de la librería junto con Openlayers.

3. Instrucciones

Para los siguientes requerimientos, enumerados según están especificados en el documento de referencia y cuya implementación requiere del uso de la librería junto con Openlayers:

Escriba "SI" en la columna "Implementado con la librería y Openlayers", si la función que cumple con el requerimiento fue efectivamente implementada con el uso de la librería y Openlayers. Escriba "NO" en caso contrario.

Escriba "SI" en la columna "Funciona", si la función opera correctamente sin ningún tipo de error o advertencia que se pueda atribuir a una incompatibilidad entre la librería y Openlayers. Escriba "NO" en caso contrario.

4. Requerimientos

N°	Requerimiento	Implementado con la librería y Openlayers	Funciona	Observaciones
1	Mostrar una capa ráster del servidor de mapas.			
4	Opción para seleccionar un elemento (feature) de la capa y mostrar su información.			
7	Opción para obtener los features de la capa y mostrarlos en el mapa.			
10	Opción para obtener el bounding box de la capa y hacer zoom sobre él.			
13	Opción para filtrar los elementos de la capa que estén dentro del área de un polígono dibujado por el usuario.			

Tabla 1. Requerimientos de la aplicación web.

Firma del evaluador

ANEXO 7. GUÍA DE OBSERVACIÓN PARA EVALUAR LA PERTINENCIA PARA EL INTERCAMBIO DE LA INFORMACIÓN

Guía de observación N° 3

1. Información de Identificación

Característica Evaluada	Pertinencia para el intercambio de información		
Evaluador			
Documento N°		**Fecha**	

2. Objetivo

Recoger información sobre los tipos de datos usados por los parámetros de entrada/salida de las funciones de Easyolmaps.

3. Instrucciones

De acuerdo a la documentación técnica y de usuario de Easyolmaps y Openlayers, Escriba "O" si el tipo de dato es de Openlayers, "E" si es de Easyolmaps y "AW" si es de alguna de las APIs Web.

4. Datos

Tipo de dato / objeto	Pertenece	Observación
GeoserverLayer		
Draw		
Style		
Coordinate		
Map		
Feature		
ProjectionLike		
Extent		
Img		
URL		

Tabla 1. Tipos de datos usados por Easyolmaps.

Firma del evaluador

ANEXO 8. GUÍA DE OBSERVACIÓN PARA EVALUAR LA SIMPLIFICACIÓN DEL USO DE OPENLAYERS

Guía de observación N° 4

1. Información de Identificación

Característica Evaluada		Simplificación del uso de Openlayers			
Evaluador					
Documento N°		**Doc. Referencia**		**Fecha**	

2. Objetivo

Recolectar datos sobre sobre el número de objetos y parámetros requeridos en la implementación de tareas específicas usando Easyolmaps y Openlayers.

3. Instrucciones

De acuerdo a las implementaciones realizadas sobre las tareas especificadas en el documento de referencia, llene las tablas con el número de objetos y parámetros requeridos en la implementación de cada tarea, teniendo en consideración lo siguiente:

- Para señalar el N° de objetos requeridos, solo considere aquellos objetos que son necesarios (como parámetros) para la creación de los objetos que permiten cumplir con la tarea especificada (objetos de la clase **Style** y **Draw** de Openlayers o sus correspondientes de Easyolmaps).

Además, anote la diferencia entre el número objetos y parámetros que se requirieron para la implementación de la tarea usando Openlayers y el número de objetos y parámetros que se requirieron usando Easyolmaps.

4. Datos

Tabla 1.Tareas asociadas a la creación de estilos para features vectoriales.

N° Tarea	**Implementación con Easyolmaps**		**Implementación con Openlayers**		**Diferencia**	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1						
2						
3						
4						
5						
6						
7						
8						

9						

Tabla 2. Tareas asociadas al dibujo de features.

N° Tarea	**Implementación con Easyolmaps**		**Implementación con Openlayers**		**Diferencia**	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1						
2						
3						

Firma del evaluador

ANEXO 9. DETALLE DE LA CAPACITACIÓN

Capacitación

1. Identificación

Tema	Desarrollo de aplicaciones web con mapas		
Instructor	Bach. en Ingeniería Informática, Henry Vanner Aquino Vegas		
Fecha	17 de agosto de 2019	**Duración**	3,5 horas

2. Objetivo

Enseñar a los asistentes, los fundamentos teóricos y las herramientas mínimas necesarias para el desarrollo de aplicaciones web de mapas, a fin de que estén aptos para evaluar la calidad de la librería.

3. Metodología

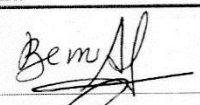
La capacitación se realizará de manera presencial y tendrá tanto parte teórica como práctica.

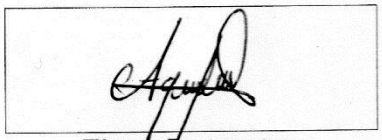
4. Contenido

- Digital Mapping, Web Mapping y Aplicaciones Web de Mapas
- Información geográfica
- Mapa
- Capas
- Web Map Service (WMS)
- Web Feature Service (WFS)
- Web Processing Service (WPS)
- GeoServer
- Openlayers
- JavaScript

5. Asistencia

El asistente debe firmar en señal de haber recibido la capacitación en su integridad, es decir, se le impartieron todos los temas que se especifican en el apartado N° 3.

Nombre y apellidos	Formación Académica	Firma
Luis Gabriel Berrú Aguilar	Ingeniero Informático.	
Jorge Andrés Acosta Pingo	Ingeniero Informático	



Firma Instructor

ANEXO 10. VALIDACIÓN DE LOS INSTRUMENTOS DE VALIDACIÓN

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Luis Gabriel Berrú Aguilar

Identificado con DNI N° 47309510 de profesión

Ingeniero Infomático ejerciendo actualmente como

Analista Programador / Desarollador Web, en la institución

EPS GRAU S.A.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 1 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			X	
Amplitud de contenido			X	
Redacción de preguntas			X	
Claridad y precisión			X	

En Piura, a los 18 días del mes de Agosto de 2019.

Firma

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Luis Gabriel Berrú Aguilar

Identificado con DNI N° 47309510 de profesión Ingeniero Informático ejerciendo actualmente como Analista Programador / Desarrollador Web, en la institución EPS GRAU S.A.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 2 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			X	
Amplitud de contenido			X	
Redacción de preguntas			X	
Claridad y precisión			X	

En Piura, a los 18 días del mes de Agosto de 2019.

Firma

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Luis Gabriel Berrú Aguilar

Identificado con DNI N° 47309510 de profesión

Ingeniero Informático ejerciendo actualmente como

Analista Programador / Desarrollador web, en la institución

EPS GRAU S.A.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 3 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			X	
Amplitud de contenido			X	
Redacción de preguntas			X	
Claridad y precisión			X	

En Piura, a los 18 días del mes de Agosto de 2019.

Firma

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Luis Gabriel Berrú Aguilar

Identificado con DNI N° 47309510 de profesión Ingeniero Informático ejerciendo actualmente como Analista Programador / Desarrollador web, en la institución EPS. GRAU. S.A.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 4 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			X	
Amplitud de contenido			X	
Redacción de preguntas			X	
Claridad y precisión			X	

En Piura, a los 18 días del mes de Agosto de 2019.

Firma

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Fabian Ary Moriño Liuiza

Identificado con DNI N° 46983363 de profesión

Ingeniero Informático ejerciendo actualmente como

analista programador web y móvil, en la institución

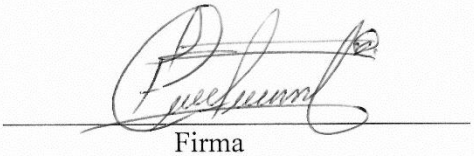
TGI Ingeniería EIRL.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 1 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			✓	
Amplitud de contenido			✓	
Redacción de preguntas			✓	
Claridad y precisión			✓	

En Piura, a los _____ días del mes de _____ de _____.



Firma

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Fabian Ary Merino Luna

Identificado con DNI N° 46983367 de profesión Ingeniero en Informática ejerciendo actualmente como analista programador web y móvil, en la institución TGI Ingeniería EIRL.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 2 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			✓	
Amplitud de contenido			✓	
Redacción de preguntas			✓	
Claridad y precisión			✓	

En Piura, a los _____ días del mes de Agosto de 2019.

Firma

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Fabian Ary Merino Livia

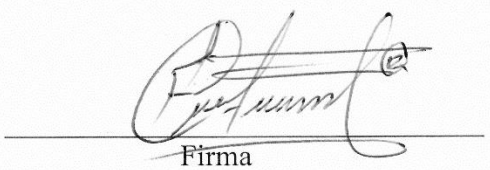
Identificado con DNI N° 46983367 de profesión Ingeniero Informático ejerciendo actualmente como analista programador web y móvil, en la institución TGI Ingeniería EIRL.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 3 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			✓	
Amplitud de contenido			✓	
Redacción de preguntas			✓	
Claridad y precisión			✓	

En Piura, a los _____ días del mes de Agosto de 2019.



Firma

CONSTANCIA DE VALIDACION DE INSTRUMENTO

Yo, Fabian Ary Merino Livia

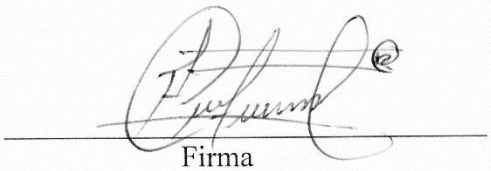
Identificado con DNI N° 46983367 de profesión Ingeniero Informático ejerciendo actualmente como analista programador web y móvil, en la institución TGI Ingeniería EIRL.

Por medio de la presente hago constar que he revisado con fines de validación del instrumento la Guía de Observación N° 4 para la evaluación de la calidad de la librería *easyolmaps*.

Luego de hacer las observaciones pertinentes, puedo formular las siguientes apreciaciones.

ITEMS	DEFICIENTE	ACEPTABLE	BUENO	EXCELENTE
Congruencia de preguntas			✓	
Amplitud de contenido			✓	
Redacción de preguntas			✓	
Claridad y precisión			✓	

En Piura, a los _____ días del mes de Agosto de 2019.



Firma

ANEXO 11. APLICACIÓN DE LAS GUÍAS DE OBSERVACIÓN

Guía de observación N°1

1. Información de Identificación

Característica Evaluada	Completitud funcional		
Evaluador	Luis Gabriel Berrú Aguilar		
Documento N°		**Fecha**	19/08/2019

2. Objetivo

Verificar el cumplimiento de los requerimientos funcionales.

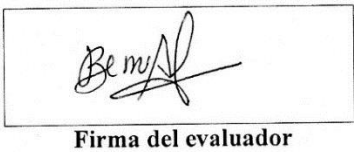
3. Instrucciones

En la columna "Cumple", escriba "SI" si la librería tiene una función que cumple con el requerimiento y si dicha función está correctamente implementada, o escriba "NO" si la librería no tiene una función que cumpla con el requerimiento o está incorrectamente implementada..

4. Requerimientos

N°	Requerimiento	Cumple	Observaciones
1	Se puede crear una capa WMS, en base a una capa de servidor WMS de GeoServer.	SI	
2	Se puede cambiar el estilo de la capa.	SI	
3	Se puede aplicar un filtro ECQL a la capa.	SI	
4	Se puede obtener features de la capa en una ubicación dada.	SI	
5	Se puede obtener la leyenda de la capa.	SI	
6	Se puede obtener features de la capa.	SI	
7	Permite la exportación de los features de la capa.	SI	
8	Se puede obtener el número de elementos de la capa.	SI	
9	Se puede obtener la descripción del tipo de feature de la capa.	SI	
10	Se puede obtener el *bounding box* de la capa.	SI	
11	Se puede ejecutar funciones de agregación a los features.	SI	

Tabla 1. Requerimientos funcionales de Easyolmaps.



Firma del evaluador

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

Guía de observación N° 2

1. Información de Identificación

Característica Evaluada	Coexistencia funcional				
Evaluador	Luis Gabriel Berrú Aguilar				
Documento N°		**Doc. Referencia**		**Fecha**	19/08/2019

2. Objetivo

Verificar en la aplicación web (referida en el documento de referencia), el correcto funcionamiento de las opciones para cuya implementación se requiere del uso de la librería junto con Openlayers.

3. Instrucciones

Para los siguientes requerimientos, enumerados según están especificados en el documento de referencia y cuya implementación requiere del uso de la librería junto con Openlayers:

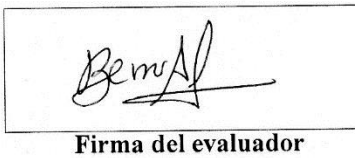
Escriba "SI" en la columna "Implementado con la librería y Openlayers", si la función que cumple con el requerimiento fue efectivamente implementada con el uso de la librería y Openlayers. Escriba "NO" en caso contrario.

Escriba "SI" en la columna "Funciona", si la función opera correctamente sin ningún tipo de error o advertencia que se pueda atribuir a una incompatibilidad entre la librería y Openlayers. Escriba "NO" en caso contrario.

4. Requerimientos

N°	Requerimiento	Implementado con la librería y Openlayers	Funciona	Observaciones
1	Mostrar una capa ráster del servidor de mapas.	SI	SI	
4	Opción para seleccionar un elemento (feature) de la capa y mostrar su información.	SI	SI	
7	Opción para obtener los features de la capa y mostrarlos en el mapa.	SI	SI	
10	Opción para obtener el bounding box de la capa y hacer zoom sobre él.	SI	SI	
13	Opción para filtrar los elementos de la capa que estén dentro del área de un polígono dibujado por el usuario.	SI	SI	

Tabla 1. Requerimientos de la aplicación web.



Firma del evaluador

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

Guía de observación N°3

1. Información de Identificación

Característica Evaluada	Pertinencia para el intercambio de información		
Evaluador	Luis Gabriel Berrú Aguilar		
Documento N°		**Fecha**	19/08/2019

2. Objetivo

Recoger información sobre los tipos de datos usados por los parámetros de entrada/salida de las funciones de Easyolmaps.

3. Instrucciones

De acuerdo a la documentación técnica y de usuario de Easyolmaps y Openlayers, Escriba "O" si el tipo de dato es de Openlayers, "E" si es de Easyolmaps y "AW" si es de alguna de las APIs Web.

4. Datos

Tipo de dato / objeto	Pertenece	Observación
GeoserverLayer	E	
Draw	E	
Style	E	
Coordinate	O	
Map	O	
Feature	O	
ProjectionLike	O	
Extent	O	
Img	AW	
URL	AW	

Tabla 1. Tipos de datos usados por Easyolmaps.

Firma del evaluador

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

Guía de observación N°4

1. Información de Identificación

Característica Evaluada	Simplificación del uso de Openlayers				
Evaluador	Luis Gabriel Berrú Aguilar				
Documento N°		**Doc. Referencia**		**Fecha**	19/08/2019

2. Objetivo

Recolectar datos sobre sobre el número de objetos y parámetros requeridos en la implementación de tare específicas usando Easyolmaps y Openlayers.

3. Instrucciones

De acuerdo a las implementaciones realizadas sobre las tareas especificadas en el documento de referencia, llei las tablas con el número de objetos y parámetros requeridos en la implementación de cada tarea, teniendo consideración lo siguiente:

- Para señalar el N° de objetos requeridos, solo considere aquellos objetos que son necesarios (con parámetros) para la creación de los objetos que permiten cumplir con la tarea especificada (objetos de clase **Style** y **Draw** de Openlayers o sus correspondientes de Easyolmaps).

Además, anote la diferencia entre el número objetos y parámetros que se requirieron para la implementación de tarea usando Openlayers y el número de objetos y parámetros que se requirieron usando Easyolmaps.

4. Datos

Tabla 1.Tareas asociadas a la creación de estilos para features vectoriales.

N° Tarea	Implementación con Easyolmaps		Implementación con Openlayers		Diferencia	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1	0	3	1	3	1	0
2	0	4	1	4	1	0
3	0	6	2	5	2	-1
4	0	7	2	7	2	0
5	0	6	3	7	3	1
6	0	4	1	4	1	0
7	0	4	1	4	1	0
8	0	6	4	10	4	4

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
CIP N° 216233

9	0	8	4	12	4	4

Tabla 2. Tareas asociadas al dibujo de features.

N° Tarea	Implementación con Easyolmaps		Implementación con Openlayers		Diferencia	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1	0	2	0	2	0	0
2	0	2	0	3	0	1
3	0	3	1	3	1	0

Firma del evaluador

LUIS GABRIEL BERRU AGUILAR
INGENIERO INFORMATICO
Reg. CIP N° 216233

Guía de observación Nº 1

1. Información de Identificación

Característica Evaluada	Completitud funcional		
Evaluador	Jorge Andrés Acosta Pingo		
Documento N°		**Fecha**	20/08/2019

2. Objetivo

Verificar el cumplimiento de los requerimientos funcionales.

3. Instrucciones

En la columna "Cumple", escriba "SI" si la librería tiene una función que cumple con el requerimiento y si dic función está correctamente implementada, o escriba "NO" si la librería no tiene una función que cumpla con requerimiento o está incorrectamente implementada..

4. Requerimientos

N°	Requerimiento	Cumple	Observaciones
1	Se puede crear una capa WMS, en base a una capa de servidor WMS de GeoServer.	SI	
2	Se puede cambiar el estilo de la capa.	SI	
3	Se puede aplicar un filtro ECQL a la capa.	SI	
4	Se puede obtener features de la capa en una ubicación dada.	SI	
5	Se puede obtener la leyenda de la capa.	SI	
6	Se puede obtener features de la capa.	SI	
7	Permite la exportación de los features de la capa.	SI	
8	Se puede obtener el número de elementos de la capa.	SI	
9	Se puede obtener la descripción del tipo de feature de la capa.	SI	
10	Se puede obtener el *bounding box* de la capa.	SI	
11	Se puede ejecutar funciones de agregación a los features.	SI	

Tabla 1. Requerimientos funcionales de Easyolmaps.

Firma del evaluador

Jorge Andrés Acosta Pingo
INGENIERO INFORMÁTICO
Reg. CIP N° 121485

Guía de observación N° 2

1. Información de Identificación

Característica Evaluada	Coexistencia funcional				
Evaluador	Jorge Andrés Acosta Pingo				
Documento N°		**Doc. Referencia**		**Fecha**	20/08/2019

2. Objetivo

Verificar en la aplicación web (referida en el documento de referencia), el correcto funcionamiento de las opciones para cuya implementación se requiere del uso de la librería junto con Openlayers.

3. Instrucciones

Para los siguientes requerimientos, enumerados según están especificados en el documento de referencia y cuya implementación requiere del uso de la librería junto con Openlayers:

Escriba "SI" en la columna "Implementado con la librería y Openlayers", si la función que cumple con el requerimiento fue efectivamente implementada con el uso de la librería y Openlayers. Escriba "NO" en caso contrario.

Escriba "SI" en la columna "Funciona", si la función opera correctamente sin ningún tipo de error o advertencia que se pueda atribuir a una incompatibilidad entre la librería y Openlayers. Escriba "NO" en caso contrario.

4. Requerimientos

N°	Requerimiento	Implementado con la librería y Openlayers	Funciona	Observaciones
1	Mostrar una capa ráster del servidor de mapas.	SI	SI	
4	Opción para seleccionar un elemento (feature) de la capa y mostrar su información.	SI	SI	
7	Opción para obtener los features de la capa y mostrarlos en el mapa.	SI	SI	
10	Opción para obtener el bounding box de la capa y hacer zoom sobre él.	SI	SI	
13	Opción para filtrar los elementos de la capa que estén dentro del área de un polígono dibujado por el usuario.	SI	SI	

Tabla 1. Requerimientos de la aplicación web.

Firma del evaluador

Jorge Andrés Acosta Pingo
INGENIERO INFORMÁTICO
Reg. CIP N° 121485

Guía de observación N°3

1. Información de Identificación

Característica Evaluada	Pertinencia para el intercambio de información		
Evaluador	Jorge Andrés Acosta Pingo		
Documento N°		**Fecha**	20/08/2019

2. Objetivo

Recoger información sobre los tipos de datos usados por los parámetros de entrada/salida de las funciones de Easyolmaps.

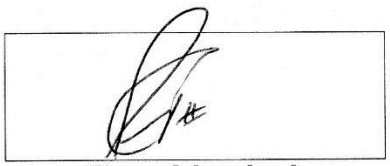
3. Instrucciones

De acuerdo a la documentación técnica y de usuario de Easyolmaps y Openlayers, Escriba "O" si el tipo de dato es de Openlayers, "E" si es de Easyolmaps y "AW" si es de alguna de las APIs Web.

4. Datos

Tipo de dato / objeto	Pertenece	Observación
GeoserverLayer	E	
Draw	E	
Style	E	
Coordinate	O	
Map	O	
Feature	O	
ProjectionLike	O	
Extent	O	
Img	AW	
URL	AW	

Tabla 1. Tipos de datos usados por Easyolmaps.



Firma del evaluador

Jorge Andrés Acosta Pingo
INGENIERO INFORMÁTICO
Reg. CIP N° 121485

Guía de observación N°4

1. Información de Identificación

Característica Evaluada		Simplificación del uso de Openlayers			
Evaluador	Jorge Andrés Acosta Pingo				
Documento N°		**Doc. Referencia**		**Fecha**	20/08/2019

2. Objetivo

Recolectar datos sobre sobre el número de objetos y parámetros requeridos en la implementación de tareas específicas usando Easyolmaps y Openlayers.

3. Instrucciones

De acuerdo a las implementaciones realizadas sobre las tareas especificadas en el documento de referencia, llene las tablas con el número de objetos y parámetros requeridos en la implementación de cada tarea, teniendo en consideración lo siguiente:

- Para señalar el N° de objetos requeridos, solo considere aquellos objetos que son necesarios (como parámetros) para la creación de los objetos que permiten cumplir con la tarea especificada (objetos de la clase **Style** y **Draw** de Openlayers o sus correspondientes de Easyolmaps).

Además, anote la diferencia entre el número objetos y parámetros que se requirieron para la implementación de la tarea usando Openlayers y el número de objetos y parámetros que se requirieron usando Easyolmaps.

4. Datos

Tabla 1.Tareas asociadas a la creación de estilos para features vectoriales.

N° Tarea	Implementación con Easyolmaps		Implementación con Openlayers		Diferencia	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1	0	3	1	3	1	0
2	0	4	1	4	1	0
3	0	6	2	5	2	-1
4	0	7	2	7	2	0
5	0	6	3	7	3	1
6	0	4	1	4	1	0
7	0	4	1	4	1	0
8	0	6	4	10	4	4

Jorge Andrés Acosta Pingo
INGENIERO INFORMÁTICO
Reg. CIP N° 121485

9	0	8	4	12	4	4

Tabla 2. Tareas asociadas al dibujo de features.

N° Tarea	**Implementación con Easyolmaps**		**Implementación con Openlayers**		**Diferencia**	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1	0	2	0	2	0	0
2	0	2	0	3	0	1
3	0	3	1	3	1	0

Firma del evaluador

Jorge Andrés Acosta Pi
INGENIERO INFORMÁTICO
Reg. CIP N° 121485

Guía de observación N° 1

1. Información de Identificación

Característica Evaluada	Completitud funcional		
Evaluador	Fabian Ary Merino Livia		
Documento N°		**Fecha**	19/08/2019

2. Objetivo

Verificar el cumplimiento de los requerimientos funcionales.

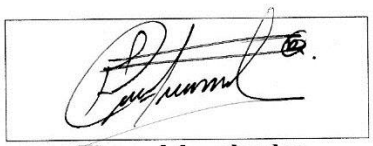
3. Instrucciones

En la columna "Cumple", escriba "SI" si la librería tiene una función que cumple con el requerimiento y si dicha función está correctamente implementada, o escriba "NO" si la librería no tiene una función que cumpla con el requerimiento o está incorrectamente implementada..

4. Requerimientos

N°	Requerimiento	Cumple	Observaciones
1	Se puede crear una capa WMS, en base a una capa de servidor WMS de GeoServer.	SI	
2	Se puede cambiar el estilo de la capa.	SI	Implementar opciones para crear estilos con la librería.
3	Se puede aplicar un filtro ECQL a la capa.	SI	
4	Se puede obtener features de la capa en una ubicación dada.	SI	
5	Se puede obtener la leyenda de la capa.	SI	Ampliar la compatibilidad para otros servidores de mapas
6	Se puede obtener features de la capa.	SI	
7	Permite la exportación de los features de la capa.	SI	
8	Se puede obtener el número de elementos de la capa.	SI	
9	Se puede obtener la descripción del tipo de feature de la capa.	SI	
10	Se puede obtener el *bounding box* de la capa.	SI	
11	Se puede ejecutar funciones de agregación a los features.	SI	

Tabla 1. Requerimientos funcionales de Easyolmaps.



Firma del evaluador

Guía de observación N° 2

1. Información de Identificación

Característica Evaluada	Coexistencia funcional				
Evaluador	Ing. Fabian Ary Merino Livia				
Documento N°		**Doc. Referencia**		**Fecha**	19/08/2019

2. Objetivo

Verificar en la aplicación web (referida en el documento de referencia), el correcto funcionamiento de las opciones para cuya implementación se requiere del uso de la librería junto con Openlayers.

3. Instrucciones

Para los siguientes requerimientos, enumerados según están especificados en el documento de referencia y cuya implementación requiere del uso de la librería junto con Openlayers:

Escriba "SI" en la columna "Implementado con la librería y Openlayers", si la función que cumple con el requerimiento fue efectivamente implementada con el uso de la librería y Openlayers. Escriba "NO" en caso contrario.

Escriba "SI" en la columna "Funciona", si la función opera correctamente sin ningún tipo de error o advertencia que se pueda atribuir a una incompatibilidad entre la librería y Openlayers. Escriba "NO" en caso contrario.

4. Requerimientos

N°	Requerimiento	Implementado con la librería y Openlayers	Funciona	Observaciones
1	Mostrar una capa ráster del servidor de mapas.	SI	SI	
4	Opción para seleccionar un elemento (feature) de la capa y mostrar su información.	SI	SI	
7	Opción para obtener los features de la capa y mostrarlos en el mapa.	SI	SI	
10	Opción para obtener el bounding box de la capa y hacer zoom sobre él.	SI	SI	
13	Opción para filtrar los elementos de la capa que estén dentro del área de un polígono dibujado por el usuario.	SI	SI	

Tabla 1. Requerimientos de la aplicación web.

Firma del evaluador

Guía de observación N°3

1. Información de Identificación

Característica Evaluada	Pertinencia para el intercambio de información		
Evaluador	Ing. Fabian Ary Menino Livia.		
Documento N°		**Fecha**	19/08/2019

2. Objetivo

Recoger información sobre los tipos de datos usados por los parámetros de entrada/salida de las funciones de Easyolmaps.

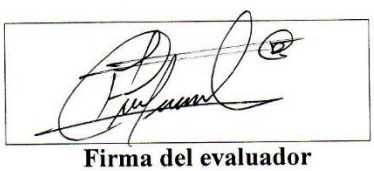
3. Instrucciones

De acuerdo a la documentación técnica y de usuario de Easyolmaps y Openlayers, Escriba "O" si el tipo de dato es de Openlayers, "E" si es de Easyolmaps y "AW" si es de alguna de las APIs Web.

4. Datos

Tipo de dato / objeto	Pertenece	Observación
GeoserverLayer	E	Extiende de la clase TileLayes de OpenLayers
Draw	E	Extiende de la clase Draw de Openlayers.
Style	E	Extiende de la clase Style de Openlayers.
Coordinate	O	
Map	O	
Feature	O	
ProjectionLike	O	
Extent	O	
Img	AW	Puede ser usada en walquier parte del DOM.
URL	AW	Se puede elegir la forma de trabajar con el dato.

Tabla 1. Tipos de datos usados por Easyolmaps.



Firma del evaluador

Guía de observación

1. Información de Identificación

Característica Evaluada		Simplificación del uso de Openlayers			
Evaluador	Ing. Fabian Ary Marino Livia.				
Documento N°		**Doc. Referencia**		**Fecha**	19/08/2019

2. Objetivo

Recolectar datos sobre sobre el número de objetos y parámetros requeridos en la implementación de tareas específicas usando Easyolmaps y Openlayers.

3. Instrucciones

De acuerdo a las implementaciones realizadas sobre las tareas especificadas en el documento de referencia, llene las tablas con el número de objetos y parámetros requeridos en la implementación de cada tarea, teniendo en consideración lo siguiente:

- Para señalar el N° de objetos requeridos, solo considere aquellos objetos que son necesarios (como parámetros) para la creación de los objetos que permiten cumplir con la tarea especificada (objetos de la clase **Style** y **Draw** de Openlayers o sus correspondientes de Easyolmaps).

Además, anote la diferencia entre el número objetos y parámetros que se requirieron para la implementación de la tarea usando Openlayers y el número de objetos y parámetros que se requirieron usando Easyolmaps.

4. Datos

Tabla 1.Tareas asociadas a la creación de estilos para features vectoriales.

N° Tarea	Implementación con Easyolmaps		Implementación con Openlayers		Diferencia	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1	0	3	1	3	1	0
2	0	4	1	4	1	0
3	0	6	2	5	2	-1
4	0	7	2	7	2	0
5	0	6	3	7	3	1
6	0	4	1	4	1	0
7	0	4	1	4	1	0
8	0	6	4	10	4	4

9	0	8	4	12	4	4

Tabla 2. Tareas asociadas al dibujo de features.

N° Tarea	**Implementación con Easyolmaps**		**Implementación con Openlayers**		**Diferencia**	
	N° de objetos (NOE)	N° de parámetros (NPE)	N° de objetos (NOO)	N° de parámetros (NPO)	NOO - NOE	NPO - NPE
1	0	2	0	2	0	0
2	0	2	0	3	0	1
3	0	3	1	3	1	0

Firma del evaluador